UNIVERSIDAD CARLOS III DE MADRID

Implementación de Algoritmos de Aprendizaje Automático para Big Data

Trabajo Fin de Grado en Ingeniería de Tecnologías de la Telecomunicación

Autor: Alba García-Tembleque Béjar

Tutor: Jerónimo Arenas García

Fecha: 21 de Junio de 2017

Índice de contenidos

Índice de tablas

Índice de figuras

Capítulo 1. Introducción

1.1 Motivación

Desde hace unos años el término “Big Data” se está colando en muchos aspectos de nuestra vida sin que nos percatemos de ello. En los medios se hace referencia a la profesión de analista de datos, en plataformas de internet como Netflix utilizan técnicas de Big Data para recomendar películas y series, etc. Pero, ¿qué hay detrás de todo esto? Moda o no, el futuro cercano del Big Data es prometedor y las inversiones que se están haciendo son cada vez más cuantiosas.

Este trabajo se enmarca dentro de este escenario de uso de datos masivos, y trata de hacer accesible a la comunidad un conjunto de herramientas para el análisis multivariante que permite su ejecución paralela sobre múltiples núcleos de computación. Este conjunto de métodos, constituyen una potente herramienta para reducir la dimensionalidad de los datos, tarea de especial importancia en los casos típicos del big data, es decir, cuando el número de datos y/o su dimensionalidad son elevados. En el presente proyecto se emplea una estrategia que permite abordar, de forma unificada, la paralelización y optimización de diversos algoritmos MVA, admitiendo soluciones regularizadas de más sencilla interpretación.

A nivel personal, este proyecto me ha permitido conocer una capa de todo lo que abarcan estas tecnologías ya que las técnicas son tan diversas como sus aplicaciones. Por ello, la realización de este proyecto ha sido una vía idónea para introducirme en el conocimiento de estas tecnologías, aprender un

lenguaje de programación nuevo (Python), y manejar herramientas de análisis de datos. Con ello se ha logrado uno de los objetivos de los Trabajos Fin de Grado: elegir en qué tema centrarnos, nuevo o no, para poder añadir el punto final a los conocimientos que nos aporta el grado.

1.2 Objetivos del Trabajo Fin de Grado

El objetivo principal de este proyecto es la implementación sobre Spark de una herramienta para análisis multivarible, que permita la optimización de los métodos de Análisis de Componentes Principales (*Principal Component Analysis*, PCA), Mínimos Cuadrados Parciales Ortonormalizados (*Orthonormalized Partial Least Squares*, OPLS) y Análisis de Correlaciones Canónicas (*Canonical Correlation Analysis,* CCA). Además de las versiones básicas de estos métodos, la herramienta está basada en un trabajo de publicación reciente que propone incorporar restricciones adicionales sobre la función de coste a fin de obtener soluciones más dispersas e interpretables. Por lo tanto, el trabajo se enmarca en el contexto del Big Data, del Aprendizaje Automático y, más concretamente, en el análisis multivariable.

Muchos de los conjuntos de datos que requieren ser procesados provienen de plataformas en las que se tienen en cuenta un gran número de variables diferentes con el fin de medir todo aquello que pueda influir a nuestros datos en cuestión. La mayoría de las veces hay un gran número de variables que no aportan nada o casi nada de información para una tarea concreta. Además muchas de ellas pueden estar correlacionadas entre sí, por lo que resultaría muy interesante eliminar estas correlaciones y las variables innecesarias, creando un conjunto nuevo de datos. Así se reduce la dimensión de los datos, lo que se traduce en una mayor rapidez del procesado y en

visualizaciones más claras, además de potenciales ventajas en términos de precisión y requisitos computacionales durante la aplicación posterior de alguna herramienta de aprendizaje automático.

En resumen, el objetivo de este proyecto es la creación de una toolbox, compuesta por sus funciones e interfaz, que aborde la implementación de versiones regularizadas de PCA, CCA y OPLS. Esta toolbox funcionará de modo distribuido por lo que se realizará sobre Spark y el lenguaje de programación Python.

Si bien el análisis de las prestaciones de los métodos en cuestión no es objeto del presente trabajo, sí se incluyen una serie de experimentos a fin de validar la correcta implementación de las técnicas e ilustrar el efecto que en las mismas tiene la variación de sus parámetros principales. Finalmente, también se quiere estudiar la escalabilidad de la implementación, para lo que se llevarán a cabo los experimentos pertinentes.

A fin de incrementar el impacto de la toolbox, se ha decidido su publicación en abierto en la plataforma GitHub. Además de la toolbox propiamente dicha, se han desarrollado una serie de notebooks de Python que la documentan y ejemplifican su uso.

1.3 Organización de la memoria

La memoria consta de siete capítulos organizados de la siguiente forma:

- Capítulo 1: Introducción del trabajo que se va a realizar y contenido de la memoria.
- Capítulo 2: Análisis del entorno del Big Data y sus principales características y descripción del planteamiento del problema que se va a llevar a cabo.
- Capítulo 3: Descripción detallada del diseño de la solución técnica donde se describen las herramientas utilizadas y el contenido de la toolbox.
- Capítulo 4: Evaluación de la toolbox mediante una serie de experimentos.
- Capítulo 5: Como ha sido la planificación del trabajo y su presupuesto previsto.
- Capítulo 6: Análisis del entorno socio-económico del proyecto
- Capítulo 7: Análisis del marco regulador en el que está enmarcado el proyecto
- Capítulo 8: Conclusiones del proyecto y trabajos futuros

Capítulo 2. Planteamiento del problema: análisis del estado del arte

En este capítulo se va a describir de forma general el entorno del Big Data, las partes que lo componen, sus características, usos y paradigmas que emplea. Por otro lado, se detallará brevemente qué es el Aprendizaje Automático (*Machine Learning*) y los problemas típicos de resolución, así como el conjunto concreto de métodos para análisis multivariable. Por último, se explicará detalladamente la formulación correspondiente a las técnicas que se van a resolver e implementar.

2.1.- ¿Qué es Big Data?

Hoy en día, Big Data es un término muy amplio que abarca desde la definición de los tipos de datos a procesar y los sistemas de almacenamiento utilizados, hasta la definición del sector TIC[1] encargado del procesamiento y análisis de datos masivos. Por otro lado, el concepto de Big Data, se relaciona con modelos de programación y sistemas distribuidos debido a la búsqueda de la rapidez en el procesamiento y obtención de los resultados. Esto se consigue gracias a la paralelización de los procesos que se explicará más adelante.

No solamente el término Big Data hace referencia a grandes cantidades de datos que exceden de la capacidad de cómputo normal, para su completa definición se habla de las cinco V que tratan de describir lo que representa:

[1] Tecnología de la información y la comunicación

- **Volumen**: Big Data está pensado para trabajar con órdenes de petabytes o más, de ahí recae la necesidad de trabajar con sistemas distribuidos. También se extrae la gran ventaja que es trabajar con estas magnitudes ya que nos permite extraer conclusiones o decidir con mucha más precisión.

- **Velocidad:** la velocidad con la que los datos son creados se materializa no solo en la necesidad del rápido procesado de éstos, sino también en la capacidad de poder recibirlos. La ventaja recae en que cuanto más actualizados estén los datos el análisis será mucho más óptimo ya que en diferentes áreas los datos son muy sensibles al paso del tiempo.

- **Variedad:** la información nos puede venir de cualquier fuente por lo que el formato que esta información tenga puede variar notablemente. Podemos definir tres tipos: datos estructurados, no estructurados y semi-estructurados.

- **Variabilidad:** la tecnología Big Data debe ser capaz de adaptarse a los cambios producidos para seguir prestando el mismo servicio.

- **Valor:** es la premisa básica que se busca, ya que lo que se pretende es que los resultados obtenidos proporcionen la mayor veracidad posible y sean útiles para su utilización.

2.1.2 Panorama actual

Trabajar con grandes cantidades de datos siempre ha sido parte del trabajo de la comunidad científica (la meteorología, la genética), o del mundo de las finanzas, entre otros. Por otro lado, los sistemas basados en sensores, sistemas de teledetección, audio etc. son cada vez más precisos y de mayor calidad, lo que conlleva más requerimiento de memoria física para almacenarlos para su posterior análisis. El gran cambio que se ha producido, siendo redundante, recae en el volumen de datos generados principalmente en internet.

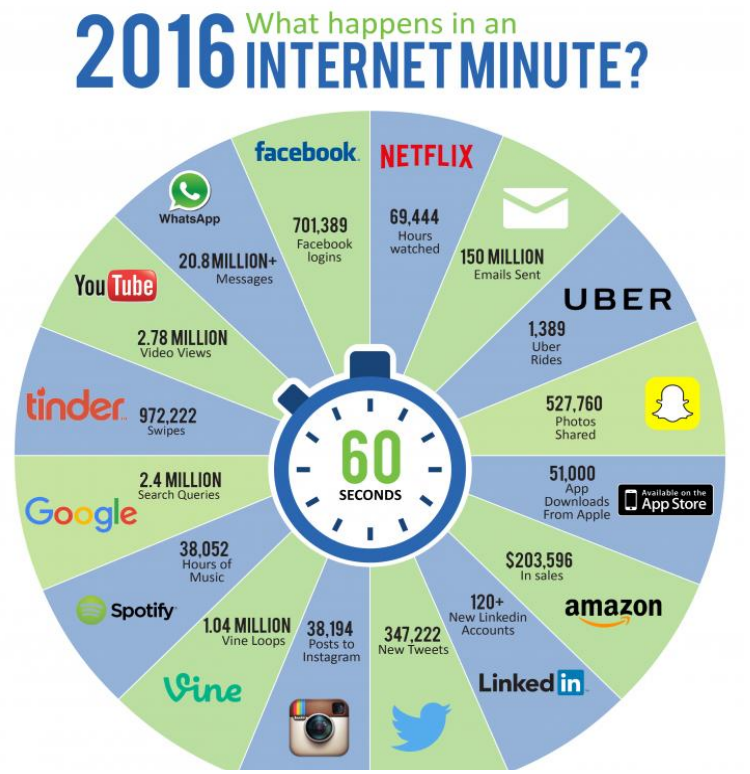


Figura 1. Datos generados en internet en un minuto, tomada de [24]

Como se puede apreciar en la Figura 1, las redes sociales generan la mayor cantidad de tránsito de datos en la web como WhatsApp, con casi 21 millones de mensajes al segundo, Facebook con unos 701,389 posts etc. Esta realidad ha llevado a muchas empresas a diseñar campañas de marketing basadas en Big Data y dirigidas a las redes sociales siendo el modo más rápido y eficaz de llegar a los usuarios.

Optar por una solución Big Data es una opción en vistas de futuro de crecimiento debido a la escalabilidad que ofrece, la forma de almacenamiento y la manera de tratar los datos. Por ello cada vez más empresas optan por invertir en esta tecnología convirtiéndola en una ventaja competitiva a la hora de llegar a los usuarios, en el tratamiento específico con cada cliente e incluso mejorando la productividad.

2.1.2 Casos de uso del Big Data

Como ya se ha mencionado anteriormente, es notable el crecimiento de aplicaciones de tecnologías de procesado de datos masivos. Entre todas ellas, las que más se pueden destacar a grandes rasgos [1], contando con que hay múltiples derivadas de cada una de ellas, son las siguientes:

- Conocer el perfil del cliente para muchas compañías se ha convertido en el principal método para ofrecer productos personalizados y predecir futuros movimientos, creando así una relación de confianza y fidelidad.

- En relación a la seguridad, el big data permite detectar patrones de conducta, posibles fraudes en cuentas bancarias, relaciones ocultas entre usuarios, etc. Analizando el tráfico de redes se pueden prevenir ataques cibernéticos, realizar análisis de datos en movimiento

detectando patrones anormales, o análisis de redes sociales para la predicción de ataques criminales etc.

- El análisis de sentimiento resulta muy importante para las operaciones de marketing. Permite predecir en tiempo real el comportamiento de los clientes, su experiencia, transacciones, así como la acogida de un producto de manera que la empresa puede actuar de forma rápida.

- El aumento de datos warehouse viene a ser el almacenamiento masivo de datos con el fin de ahorrar tiempo y costes para el futuro. Para mayor eficiencia, se deben diferenciar aquellos datos que pueden ser más antiguos pero siguen teniendo validez de aquellos más nuevos que no aportan información.

2.1.3 Arquitectura

La arquitectura para un sistema de análisis de datos que puede ser referenciado para un sistema Big Data, cuenta con cuatro componentes principales y bien diferenciados que son la recolección de los datos, su almacenamiento, el procesamiento y análisis de ellos y la visualización. Hay que tener en cuenta que la arquitectura Big Data no es un estándar y esta puede ser una aproximación.

- *Recolección de datos: Hoy en día se pueden obtener datos de una gran variedad de sitios y en diversos formatos; por ejemplo, documentos, bases de datos, redes sociales etc. En cuanto a la naturaleza de los datos, conviene distinguir los dos tipos siguientes:*
    - *Streaming: cuando la información es generada o transmitida de forma continua, como un flujo de datos.*
    - *Batch o por lotes: cuando los datos se generan o actualizan cada cierto tiempo.*

- Almacenamiento: El modo de almacenamiento ha tenido que cambiar debido al Big Data, ya que muchos de los datos generados no cuentan con una estructura definida y son variables en comparación con las Bases de Datos relacionales de uso tradicional. Para datos no estructurados se recurre a bases de datos no relacionales (noSQL) como MongoDB o Cassandra.

- Procesamiento y análisis: En este paso se pretende dar valor a todos aquellos datos que hemos recolectado extrayendo información útil de ellos, procesándolos y analizándolos con las diferentes técnicas de las que se dispone, principalmente técnicas de Inteligencia de Negocio y/o Aprendizaje Automático. No importa la cantidad de datos de la que dispongamos si no sabemos cómo trabajar y obtener conocimiento de ellos.

- Visualización: es una capa muy importante ya que en muchas ocasiones es muy complicado dar interpretabilidad a los datos. Por ello existen métodos que facilitan la creación de gráficas, mapas etc. que proporcionan visibilidad y permiten entender el valor intrínseco de los datos analizados.

2.2 Computación en paralelo

La computación en paralelo y los paradigmas que hacen uso de ella, se han postulado como los métodos más eficientes para el tratamiento de grandes cantidades de datos.

La computación en paralelo se define como la división de una tarea en varias subtareas que se ejecutan en paralelo. Se basa en el principio de que problemas grandes pueden resolverse más rápidamente si se dividen en problemas más pequeños [2]. Se pueden distinguir entre diferentes tipos de computación paralela:

- **Paralelismo a nivel de bit:** donde dependiendo del tamaño de la palabra de un procesador las instrucciones que tiene que realizar son más o menos.
- **Paralelismo a nivel de instrucción:** que hace referencia a las instrucciones de un programa que pueden reordenarse y ser ejecutadas de forma paralela.
- **Paralelismo de datos:** es la distribución de los datos en diferentes nodos, de manera que cada nodo se ocupa de un subconjunto diferente de los datos, y los nodos pueden trabajar en paralelo.
- **Paralelismo de tareas:** consiste en tratar los datos, homogéneos o heterogéneos, realizando tareas completamente diferentes para cada uno de ellos.

La dificultad de la computación en paralelo viene de que tradicionalmente el software se ha escrito de forma secuencial, de forma que las instrucciones representan la secuenciación de ejecución en la CPU. Para el desarrollo de algoritmos paralelos existen una serie de paradigmas que aportan estrategias para la resolución de estos problemas. En la siguiente subsección se explican el paradigma que más implicaciones ha tenido en cuanto a la computación en paralelo y al entorno Big Data.

2.2.1 MapReduce

MapReduce es un modelo de programación creado por dos ingenieros de Google mucho antes de que se empezara a popularizar el término Big Data. Este paradigma nace a partir de la necesidad que tenía Google de procesar grandes volúmenes de datos y de desarrollar un esquema paralelizable con el fin de reducir el tiempo de procesamiento. Más adelante este paradigma se comienza a popularizar y muchas aplicaciones lo incluyen en sus soluciones. La más famosa es Apache Hadoop [3], siendo quizá ésta la primera aplicación que puede considerarse dentro del paradigma de Big Data.

MapReduce proporciona procesamiento de datos en paralelo y utiliza un sistema de archivos distribuidos (*Hadoop Distributed File System, HDFS*). Este paradigma está concebido para trabajar con tamaños de datos del orden de gigabytes y terabytes. Como su nombre indica, la arquitectura consta de dos partes: las funciones *Map* y *Reduce*, que se ejecutan distribuidamente en los nodos que componen su estructura:

- Map: esta función recibe los datos que previamente han sido divididos por una función *split* automática. Posteriormente "mapea" los datos según se quiera obteniendo una pareja clave/valor.

$$\text{Map: } (\text{clave}, \text{valor1}) \rightarrow \text{lista } (\text{clave}, \text{valor2})$$

La función Map supone una transformación de cada bloque de datos, según una función seleccionada o implementada por el usuario. Dichas tareas pueden realizarse de forma independiente para cada bloque de datos, lo que posibilita la ejecución en paralelo en los distintos nodos de computación disponibles.

- Reduce: esta función recibe como parámetros una clave y una lista de los valores asociados a esa clave. Su salida corresponde a la combinación de esos valores de cualquier forma (suma, máximo, etc.)

$$\text{Reduce: } (\text{clave}, \text{lista}(\text{valor})) \rightarrow (\text{clave}, \text{valor\_nuevo})$$

En la Figura 2 podemos ver cómo se realiza el proceso con un ejemplo sencillo, consistente en contar el número de apariciones de cada letra en una lista desordenada. Las claves de nuestros datos de entradas son las letras, se realiza una función split para separarlas. En la fase Map se asignan los valores, en este caso el número uno. Después de esta fase automáticamente se ordenan las salidas del mapeo. Por último la fase Reduce realiza, en este caso, la suma de todos los valores correspondientes a la misma clave, resultando así ser un contador de letras.

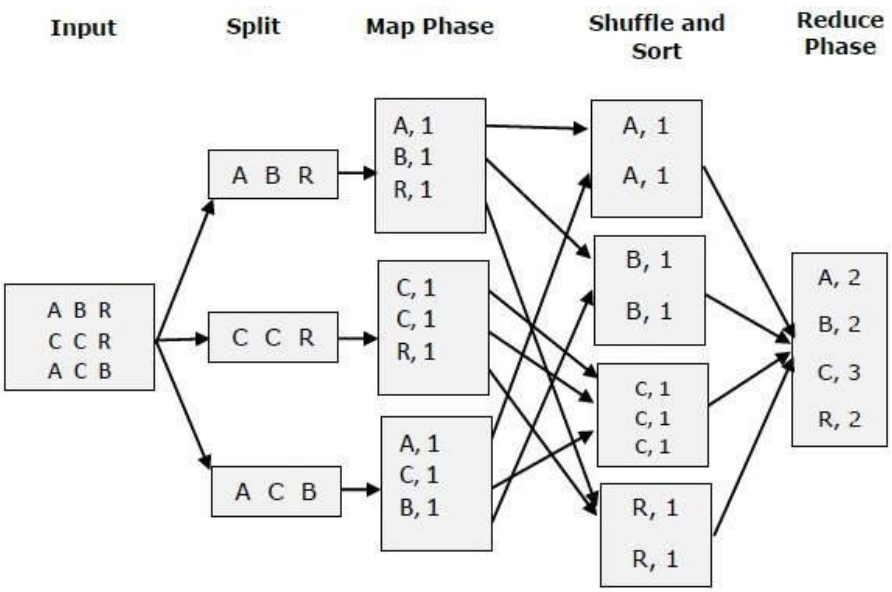


Figura 2. Contador de letras con MapReduce [4]

2.3 Aprendizaje automático

El aprendizaje automático (en inglés, *Machine Learning)* es una rama dentro del campo de la Inteligencia Artificial relacionada con la estadística [5]. Este campo intenta dotar a los ordenadores o máquinas la habilidad de aprender sin haber sido específicamente programados y actuar de acuerdo a esa información aprendida. Este aprendizaje se hace posible detectando patrones a partir de conjuntos de datos para la creación de un modelo predictivo.

Las aplicaciones del Aprendizaje máquina son muy diversas. A modo de ejemplo:

- Predicción: donde se obtienen unas variables de salida de acuerdo a modelos previamente aprendidos. Por ejemplo para predecir el tráfico, el tiempo etc.

- Minería de datos: es una de las más importantes, donde el objetivo es extraer patrones de información relevante de grandes conjuntos de datos.

- Reconocimiento de imágenes: por ejemplo de rostro, de imágenes médicas etc.

- Análisis de comportamiento de consumo y productividad: donde se analiza a los clientes y sus consultas para ofrecer el producto que pueda interesarle.

Típicamente la construcción de los modelos se apoya en la disponibilidad de un conjunto de datos de entrenamiento, que suele consistir en

un conjunto de patrones que es la unidad para llevar a cabo la tarea de predicción, recomendación, etc. Según se disponga o no de la salida deseada del modelo para dichos datos de entrenamiento distinguimos:

- Aprendizaje supervisado: Estos algoritmos parten de un conocimiento previo a través de unos datos de entrenamiento en los que se conoce una serie características y sus correspondientes etiquetas (para un problema de Clasificación) o la respuesta para esos datos (para un problema de Regresión). El objetivo es construir un modelo que generalice correctamente para datos diferentes a los presentes durante el entrenamiento, es decir, que tenga una buena capacidad predictiva para los datos que vayan a presentarse en el futuro.

- Aprendizaje no supervisado: En este tipo de aprendizaje no disponemos de etiquetas para los datos de entrenamiento, por lo que estos algoritmos tratan simplemente de buscar patrones o estructuras en los datos.

2.3.1 Tipos de problemas

Según la naturaleza de la tarea que se desea llevar a cabo, distinguimos distintos tipos de algoritmos de aprendizaje automático:

2.3.1.1 Clasificación

Este tipo de problemas consiste en asignar a un vector de p observaciones $X = (x1, x2, x3, \ldots, x_n)$ a una de entre m clases $posibles$: $C1, C2, C3, \ldots C_m$. Si $m = 2$ se denomina clasificación binaria, si $m > 2$ hablamos de clasificación multiclase.

Utilizando el ejemplo de diagnosis médica, supongamos conocidas las probabilidades a priori de las clases P(C1)= 0,7 que supone estar enfermo y P(C2)= 0,3 que supone no estarlo. El vector x en este caso constaría de diferentes medidas tales como la temperatura. Para determinar a qué clase pertenece cada individuo se calculan las probabilidades a posteriori P(C1|x) y P(C2|x). La regla de decisión vendrá determinada por asignar el vector x a la clase con mayor probabilidad a posteriori. En este caso, el algoritmo de aprendizaje automático debe entrenarse a partir de un conjunto de pacientes previamente etiquetados. Nótese, además, que en la mayoría de los casos la probabilidad de error no podrá disminuirse a valores arbitrariamente bajos, ya que posiblemente las poblaciones de datos de ambas clases se encuentren solapadas.

2.3.1.2 Regresión

Con este tipo de problemas lo que se intenta es buscar una relación matemática entre dos o más variables, si la hay, para que en futuro permita el pronóstico del valor que esas variables van a tomar. La diferencia principal con el problema de clasificación, consiste en que la variable que se desea estimar puede tomar valores en uno o varios intervalos de números reales. El objetivo principal es encontrar una función matemática $\mathrm{y} = \mathrm{f}(\mathrm{x})$. La variable x puede ser un número o un vector de variables $\mathrm{y} = \mathrm{f}(\mathrm{x1}, \mathrm{x2}, \ldots \mathrm{x_m})$ lo que se conoce como funciones de variables vectoriales. También existen funciones vectoriales que proporcionan varias salidas reales: $(\mathrm{y1}, \mathrm{y2}, \ldots, \mathrm{yd}) = \mathrm{f}(\mathrm{x1}, \mathrm{x2}, \ldots \mathrm{x_m})$. La función matemática se resuelve mediante un problema de mínimos cuadrados el cual intenta minimizar el error cuadrático medio resolviendo la siguiente ecuación:

$$\sum_{k=1}^{n} (y_k - \sum_{j=1}^{m} w_j f_j(x_k))^2$$

2.4 Análisis multivariable, MVA

El análisis multivariable se ocupa de analizar las relaciones existentes entre las variables que caracterizan la representación de un conjunto de datos y/o las variables objetivo. El hecho de tener varias variables de medida proporciona un mejor entendimiento del problema y facilita la construcción de un modelo más exacto. Este análisis consta de métodos para la extracción de variables disponibles con la menor pérdida de información, explotando la existencia de correlaciones entre variables, y resultando habitualmente en una reducción significativa de la dimensionalidad de los datos. Los métodos más comunes y que son objeto de estudio en este trabajo son los siguientes: Análisis de Componentes Principales (*Principal Component Analysis*, PCA), Análisis de Correlaciones Canónicas (*Canonical Correlation Analysis*, CCA) y Mínimos Cuadrados Parciales Ortonormalizados (*Orthonormalized Partial Least Squares*, OPLS). Estos métodos poseen la propiedad de que las variables extraídas no poseen correlación entre ellas, lo que facilita la selección de un subconjunto de variables extraídas, y facilita la aplicación posterior de otras técnicas de aprendizaje automático.

2.4.1 Análisis de Componentes Principales (PCA)

Esta técnica reduce la dimensión de los datos proyectándolos sobre un nuevo sistema de coordenadas. Se construye una transformación lineal que nos lleva a este sistema de coordenadas para los nuevos datos donde la varianza de mayor tamaño se captura en el primer eje, la segunda más grande en el segundo eje y así sucesivamente, garantizando además que los sucesivos vectores de proyección son perpendiculares entre sí. Esta base de transformación depende de las observaciones y se forma a partir de los autovalores más significativos de la matriz de covarianza de los datos, que representan las componentes principales [6].

2.4.2 Análisis de Correlaciones Canónicas (CCA)

CCA explota las relaciones lineales entre dos vectores multidimensionales de datos de entrada y salida. Consiste en encontrar dos bases, una para x y otra para y, tal que la matriz de correlaciones entre las dos variables es diagonal y las correlaciones de la diagonal se maximizan. Una de las propiedades importantes de las correlaciones canónicas es que no varían con transformaciones afines de las variables, lo que hace que este método sea robusto frente al reescalado de las variables [7].

2.4.3 Mínimos cuadrados parciales ortonormalizados (OPLS)

OPLS puede considerarse una variante de CCA en la que tiene en cuenta la varianza de las variables objetivos (pero no de la representación de entrada de los datos). Puede demostrarse que este método encuentra la representación óptima de los datos de entrada, para una dimensionalidad dada, en términos de minimización del error cuadrático de aproximación de los datos de salida.

OPLS obtiene un modelo de regresión lineal proyectando las variables predichas y las observadas en nuevo espacio. Así se intenta explicar la dirección de máxima varianza en el espacio de Y con el sentido multidimensional en el espacio de X. Es especialmente usado en el caso donde número de variables independientes es significativamente más grande que el número de datos de los que disponemos.

Al contrario que PCA que considera únicamente las covarianzas entre las variables de entrada, CCA y OPLS utilizan las etiquetas de los datos de entrenamiento para encontrar la proyección más adecuada del espacio de entrada, lo que hace que estos métodos obtengan generalmente mejores prestaciones en problemas de clasificación y regresión [8].

2.5 Descripción del problema a implementar

Una propiedad relevante de los métodos de análisis de múltiples variables mencionados es que su resolución implica principalmente el cálculo de matrices de covarianzas de datos de entrada y salida, siendo esta una tarea fácilmente paralelizable bajo el paradigma MapRedude. Por tanto, el presente trabajo aborda la implementación de una toolbox para estos métodos sobre Spark.

Concretamente, seguiremos la implementación de estos métodos sugerida en [9], que presenta las siguientes ventajas respecto a la formulación y optimización clásica de estos métodos:

- Emplea un marco unificador para PCA, CCA y OPLS, de manera que la implementación se simplifica y permite al usuario abordar únicamente uno u otro método mediante la definición de matrices auxiliares específicas a cada método.
- Permite la incorporación de restricciones basadas en las normas l_1 y $l_{2,1}$ de los vectores de proyección. El uso de estas restricciones favorece la aparición de ceros en los vectores de proyección, y favorecen la interpretabilidad de la solución. Para la incorporación de restricciones, el método se basa en la combinación de un problema de mínimos cuadrados regularizados para la extracción de las características de entrada y otro de autovalores generalizados.
- La técnica descrita en este trabajo proporciona una alternativa a otras implementaciones previas que se caracteriza por conseguir una mayor decorrelación de las variables proyectadas, una propiedad deseable para cualquier método de análisis de múltiples variables.

Revisamos brevemente en lo que queda de capítulo el método propuesto en [9] para la resolución de los problemas PCA, CCA y OPLS regularizados. El objetivo de esta descripción es simplemente proporcionar una

mínima enumeración de los pasos que requiere el algoritmo de optimización, y que serán implementadas en la toolbox objeto de este trabajo fin de grado. La revisión detallada del método, así como la demostración de la convergencia a la solución deseada queda fuera del ámbito de esta memoria, y se recomienda al lector interesado la consulta de [9] para detalles acerca de las mismas.

La solución que se va a plantear persigue la minimización de la siguiente función objetivo:

$$L(W,U) = \left|\Omega^{\frac{1}{2}}(Y - WU^T X)\right|^2 + \gamma R(U)$$
$$= Tr\{Y^T \Omega \mathrm{Y}\} - 2Tr\{U^T C_{XY} \Omega \mathrm{W}\} + Tr\{U^T C_{XX} U W^T \Omega \mathrm{W}\} + \gamma R(U) \qquad (1)$$

donde $X = [X_1, \dots, X_N]$ e $Y = [Y_1, \dots, Y_N]$ son las matrices de datos de entrada y salida respectivamente. $C_{XX} = XX^T$, $C_{YY} = YY^T$ y $C_{XY} = XY^T$ corresponden a las matrices de covarianza de las entradas, salidas, y varianzas cruzadas entre las entradas y salidas, respectivamente. La matriz Ω se particulariza para cada método: $\Omega = C_{YY}^{-1}$ para CCA, $\Omega = I$ para OPLS y CCA. Además, para PCA ha de emplearse $X = Y$ para PCA. $R(U)$ es el término de regularización (más adelante se explicará los tipos se pueden utilizar), y γ el parámetro utilizado para ajustar la relevancia del término de regularización. Finalmente, W se considera la matriz con los coeficientes de la regresión lineal, mientras que la matriz U, es la matriz de proyección que permite extraer las proyecciones de los datos de entrada como $X' = U^T X$.

Nótese que en (1), |·| representa la norma de Frobenius, y que su minimización ha de llevare a cabo bajo la restricción de incorrelación de las variables extraídas, i.e., $\mathrm{U^T C_{XX} U} = \Delta$, siendo Δ una matriz diagonal.

Primeramente, este problema se resuelve para el caso de que el término $R(U)$ sea derivable. Aplicando la restricción $W^T \Omega W = I$, que puede comprobarse que es equivalente a la de decorrelación de las variables proyectadas, y derivando respecto a U la ecuación (1), se obtiene una posible

solución de U en función de W la cual después de varias transformaciones [9] conduce a un problema estándar de autovalores. En la Tabla 1 se muestran el problema de autovalores concreto que ha de resolverse para cada uno de los métodos:

	V (eig. problem)	W	U
CCA	$\mathbf{C_{YY}^{-\frac{1}{2}} C_{XY}^{\top} \tilde{C}_{XX}^{-1} C_{XY} C_{YY}^{-\frac{1}{2}} V = V\Lambda}$	$\mathbf{W = C_{YY}^{\frac{1}{2}} V}$	$\mathbf{\tilde{C}_{XX}^{-1} C_{XY} C_{YY}^{-\frac{1}{2}} V}$
OPLS	$\mathbf{C_{XY}^{\top} \tilde{C}_{XX}^{-1} C_{XY} V = V\Lambda}$	$\mathbf{W = V}$	$\mathbf{\tilde{C}_{XX}^{-1} C_{XY} V}$
PCA	$\mathbf{C_{XX}^{\top} \tilde{C}_{XX}^{-1} C_{XX} V = V\Lambda}$	$\mathbf{W = V = U}$	$\mathbf{\tilde{C}_{XX}^{-1} C_{XX} V}$

Tabla 1. Problema de autovalores particularizado para CCA, OPLS y CCA. Tomado de [9].

Es importante remarcar que en ausencia de regularización esta aproximación sigue produciendo variables incorreladas, y constituye una solución en un único paso al problema planteado.

En segundo lugar, se propone una solución iterativa para el caso en que los tipos de regularización no sean derivables, por ejemplo la regularización LASSO o la norma $l_{2,1}$, las cuales se explicaran más adelante con detalle.

El proceso iterativo consta de dos pasos básicos:

- Paso- U. Consta de un problema básico de mínimos cuadrados, suponiendo fija la matriz W (satisfaciendo $W^T \Omega W = I$). Para encontrar la matriz U que minimiza la siguiente expresión:

$$|| Y' - U^T X ||^2 + \gamma R(U), \quad (2)$$

con $Y' = W^T \Omega \mathrm{Y}$, emplearemos las implementaciones de Elastic Net disponibles sobre Spark en MLLib.

- Paso-W. Suponiendo U fijada, buscaremos la matriz W que minimiza (1) sujeto a $W^T \Omega W = I$ o, reescribiendo este paso en términos de $V = \Omega^{1/2} W$ resolver V minimizando :

$$| |\bar{Y} - VX'||^2, \text{ sujeto a } V^T V = I, \quad (3)$$

donde $\bar{Y} = \Omega^{\frac{1}{2}} Y$.

En la literatura relevante, el paso W normalmente ha sido resuelto como un problema ortogonal de Procustes. Este tipo de problemas consiste en encontrar una matriz ortogonal que mapee lo más próximo posible dos matrices: $R = argmin_{\Omega} | |\Omega \mathrm{A} - \mathrm{B}||_{\mathrm{F}}$ sujeto a $\Omega^T \Omega = \mathrm{I}$. Pero el inconveniente de esta solución es que ignora la no restricción de no correlación entre las variables extraídas.

La solución propuesta en [9], que será la adoptada en nuestra implementación, consiste en resolver el paso W como un problema de autovalores. Reformulando (3) usando multiplicadores de Lagrange, y derivando esta nueva expresión, se llega a los problemas de autovalores recogidos en la Tabla 2.

	$\mathbf{U}$-step (reg. LS)	$\mathbf{W}$-step (eigenvalue problem)
reg. CCA	$\arg\min_{\mathbf{U}} \|\mathbf{Y}' - \mathbf{U}^\top \mathbf{X}\|_F^2 + \gamma R(\mathbf{U})$	$\mathbf{C}_{\mathbf{YY}}^{-\frac{1}{2}} \mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}\mathbf{U}^\top \mathbf{C}_{\mathbf{XY}} \mathbf{C}_{\mathbf{YY}}^{-\frac{1}{2}} \mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$
reg. OPLS	$\arg\min_{\mathbf{U}} \|\mathbf{Y}' - \mathbf{U}^\top \mathbf{X}\|_F^2 + \gamma R(\mathbf{U})$	$\mathbf{C}_{\mathbf{XY}}^\top \mathbf{U}\mathbf{U}^\top \mathbf{C}_{\mathbf{XY}} \mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$
reg. PCA	$\arg\min_{\mathbf{U}} \|\mathbf{X}' - \mathbf{U}^\top \mathbf{X}\|_F^2 + \gamma R(\mathbf{U})$	$\mathbf{C}_{\mathbf{XX}}^\top \mathbf{U}\mathbf{U}^\top \mathbf{C}_{\mathbf{XX}} \mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$

Tabla 2. Pasos W y U para la optimización de PCA, CCA y OPLS regularizados. Tomada de [9].

Cuando la propiedad de no correlación entre variables proyectadas se consigue, el proceso de extracción de características ofrece múltiples ventajas:

- Los problemas de mínimos cuadrados son independientes sobre cada dimensión y los efectos de variación de los datos de entrada se aíslan en cada dirección.

- La selección óptima del subconjunto de características se reduce a la selección de las características con mayor autovalor asociado.

Capítulo 3.
Implementación de una toolbox para análisis multivariable

En este capítulo se detallarán las herramientas que se han utilizado en el diseño e implementación de la toolbox de análisis de múltiples variables y los tipos de regularizaciones, normalizaciones y estructuras de datos que soporta. También se explicará la estructura de la toolbox con las partes que la componen y su funcionamiento.

3.1 Soluciones tecnológicas empleadas

En esta sección se explican las herramientas que se van a utilizar para la implementación técnica.

3.1.1 Apache Spark

Su nacimiento surge en los laboratorios AMPLab en la Universidad de Berkeley. En 2014 fue acogido como un proyecto “Top-Level” de la Apache Software Foundation y nació la compañía Databricks para dar soporte a su desarrollo.

Apache Spark es un framework open source de procesamiento paralelo en clústeres diseñado para ser rápido y de propósito general. La rapidez la consigue ejecutando en memoria de manera distribuida, frente a otras soluciones que están basadas en el intercambio de datos con persistencia en disco duro. Spark también extiende el modelo MapReduce. Está diseñado para

trabajar con grandes cantidades de datos de diversas fuentes, proporcionando APIs de alto nivel en Java, Scala y Python, las cuales facilitan su uso.

Para el desarrollo de mi implementación he utilizado esta plataforma web, Databricks, ya que proporciona un cluster autoescalable de Apache Spark, notebooks para Python con colaboración en tiempo real e integración GitHub, despliegue en un click para visualizar los Spark Jobs y muchas más utilidades.

Los principales módulos de los que consta Spark son [10]:

- Spark SQL para el procesamiento de datos estructurados y realizar consultas SQL. Proporciona una abstracción denominada DataFrame que se define como una colección de datos distribuida organizada en columnas.

- GraphX que permite realizar computación paralela de grafos

- Spark Streaming que proporciona procesamiento de flujo de datos en tiempo real escalable. Proporciona una abstracción denominada DStream, que consiste en una secuencia continua de RDDs, la abstracción básica de Spark que se explicarán más detalladamente.

- MLlib que es una librería para aprendizaje automático. Proporciona algoritmos de aprendizaje básicos de regresión, clasificación, clustering, modelado de tópicos, etc.

La base de Spark es Spark Core, la cual contiene la funcionalidad básica de recuperación de fallos, planificación de tareas, el API que define los RDDs etc.

Apache Spark ofrece también funciones para acceder a diversas fuentes de datos además de bases de datos SQLP. Así, puede importar datos de un sistema de archivos distribuido para Hadoop (HDFS), acceder a fuentes de datos noSQL como Cassandra, importar ficheros json, y muchos más como puede verse en la Figura 3.

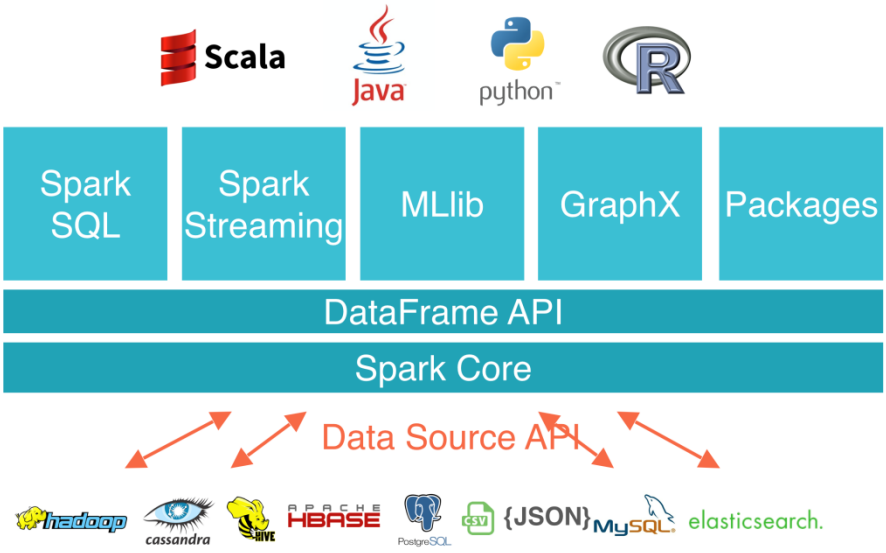


Figura 3. Estructura de los módulos Spark .Tomada de [11]

La abstracción principal que usa Spark, y la que se va a utilizar para el diseño de la toolbox de este trabajo se denomina RDD (Resilient Distributed Dataset), y representa una colección de elementos en memoria distribuidos en particiones. Cada nodo del cluster puede contener una o varias particiones y operar de forma paralela con estos datos. Estas colecciones son inmutables, al realizar transformaciones sobre ellos se están creando nuevos RDD.

Hay varias formas de crear un RDD:

- Paralelizando una colección que esté almacenada en memoria como puede ser un array o una lista con el método *parallelize()*.

- Cargando un archivo de datos con múltiples formatos como puede ser un fichero de texto, en formato csv[2], etc. Para crear un RDD a partir de un fichero de texto se puede emplear el método *textFile()*.

- Aplicando una transformación en un RDD para crear otro.

En Spark se pueden realizar dos tipos de operaciones sobre los RDD:

- Transformaciones: son las operaciones que dan como resultado la creación de un nuevo RDD. Son ejecutadas de forma *lazy* que significa que se ejecutan únicamente cuando sobre ellas se utiliza una acción. La ventaja de la evaluación *lazy* es la de agrupar operaciones y retrasar la ejecución al momento en que se require obtener los resultados. Las funciones de transformación más sencillas son *map()* y *filter().*

- Acciones: estas operaciones dan un valor final como resultado después de evaluar el RDD aplicando operadores "reduce". Las acciones fuerzan a la evaluación de las transformaciones. Algunas de estas operaciones son: *reduce()* y *count().*

Es importante destacar la inclusión de DataFrames en la API de MLlib. La abstracción de DataFrames previamente se había definido dentro de la API SparkSQL describiéndose como conjuntos de datos distribuidos y organizados por columnas, conceptualmente equivalente a una tabla de base de datos relacional. Esta API está inspirada en los dataframes de R y Python (Pandas) con el fin de aprovechar el poder del procesado distribuido para Big Data. También son evaluados de forma *lazy* como los RDDs. Pueden ser

[2] CSV: Formato sencillo de documentos para representar tablas de datos donde las columnas se separan por comas.

construidos a través de numerosas fuentes y soportando un amplio rango de los formatos más populares.

Tal ha sido el éxito y repercusión de esta nueva abstracción que la principal API de Aprendizaje Automático para Spark es ahora la API de DataFrames dentro del paquete de Spark ML. Este cambio es debido a que esta API es mucho más fácil de usar para el usuario, está provista de una mayor uniformidad con algoritmos de ML y con otros lenguajes. Además, DataFrames incluye los beneficios de poder realizar consultas SQL, el uso de optimizadores, etc [12].

Esta actualización implica que, aunque MLlib seguirá soportando la API de RDDs, actualmente se realiza únicamente el mantenimiento de la API basada en RDDs, pero la inclusión de nueva funcionalidad se hará únicamente para la API basada en Dataframes, si bien a fecha de presentación de este trabajo no ha sido todavía migrada toda la funcionalidad que aportaba la API anterior. Se espera que en el momento que la API de DataFrames alcance las características que hoy en día asegura la API de RDDs, ésta será definitivamente desaprobada para la versión 2.2 de Spark y eliminada para la versión 3.0. [13]

Los DataFrames son el futuro de Spark por lo que en relación al diseño de la toolbox de MVA, la exportación del código para usar Dataframes en lugar de RDDs constituye una línea de trabajo inmediata para la continuación del presente trabajo de Fin de Grado.

3.1.2 Python

Los lenguajes soportados por Spark son Scala, Java y Python. Para la implementación de este Trabajo de Fin de Grado se ha utilizado Python por numerosas razones. Python es un lenguaje de programación con una sintaxis legible y sencilla, muy cercana al lenguaje natural, lo cual es su principal ventaja. Su estructura es bastante rígida, evitando el uso del punto y coma y llaves ya que se basa en la indentación, que no es opcional. Permite reducir considerablemente la extensión del código, en comparación con con otros lenguajes que necesitan una estructura mucho más compleja. No es un lenguaje fuertemente tipado, ya que el tipo de dato se determina dependiendo del valor asignado. Cuenta con una gran cantidad de librerías desde manipulación de imágenes a cálculos científicos.

La API que permite utilizar Spak con Python se denomina PySpark. Cuenta con paquetes para todos los módulos de Spark (SQL, Streaming, ML y MLlib). Las clases públicas que la componen son [14]:

- *SparkContext()* que es la clase principal que permite la conexión con el cluster de Spark y la creación de RDDs.

- *RDD()* que contiene las operaciones básicas para poder trabajar con esta abstracción.

- *Broadcast()* que permite leer variables desde el cluster devolviendo un objeto para lectura distribuida.

- *Acumulator()* ofrece la posibilidad de realizar la operación conmutativa y asociativa de la suma directamente en el cluster de Spark.

- *SparkConf()* usada para la configuración de la aplicación de Spark, como modificaciones de parámetro en pares de clave-valor.

- *SparkFiles()* proporciona las rutas de directorios y rutas absolutas de ficheros.

- *StorageLevel()* contiene las alertas básicas de control de memoria de un RDD.

3.1.3 Librerías y APIs

Las librerías de las que se ha hecho uso en el diseño de la toolbox de análisis de múltiples variables son las siguientes:

- **MLlib** como se ya se ha indicado anteriormente es la librería de Machine Learning de Spark; es una librería escalable y de fácil uso. Son múltiples los algoritmos que incluye para: Clasificación, Regresión, Árboles de decisión, Clustering, Topic Modeling, etc. También proporciona utilidades para la transformación de características (estandarización, normalización...), construcciones de Pipeline, operaciones de álgebra lineal y estadística, etc.

- La librería **NumPy** opera internamente con MLlib para Python en Spark. Esta librería ofrece el fundamental paquete de cálculo científico para Python el cual contiene la creación de objetos multidimensionales (vectores y matrices), funciones para álgebra lineal, transformaciones de

Fourier; en resumen una gran biblioteca de cálculo entre objetos de alto nivel.

- **Scikit-Learn** también es una librería Open source de Machine Learning en Python, construida directamente sobre otra librería de Python: SciPy (proporciona el uso de cálculos científicos). Ofrece herramientas simples y eficientes para procesos de data mining y análisis de datos. Cuenta también con algoritmos para clasificación, regresión, clustering, reducción de dimensión, selección de modelos y preprocesado. Es una librería con una documentación y ejemplos de alta calidad y con expectativas de crecimiento muy favorables.

3.2 Tipos de regularización

El objetivo básico de usar técnicas de regularización es introducir restricciones adicionales para evitar el sobreajuste del modelo y/o mejorar la interpretabilidad de la solución. En técnicas predictivas, el sobreajuste se da cuando un modelo es muy complejo o consta de muchos parámetros en relación a las observaciones de las que disponemos, lo que nos lleva a perder la capacidad de generalización del modelo ya que no reaccionará adecuadamente a variaciones en los datos de test que no hayan sido observadas en los datos de entrenamiento.

Una de las familias de regularizadores es la correspondiente a la norma l_p. En esta sección se centra la descripción de los regularizadores que soporta MLlib: normas L1, L2 y elastic net.

3.2.1 Regularizadores en MLlib

Estos tipos de normas se aplican habitualmente como penalización a problemas de regresión, normalmente a problemas de mínimos cuadrados. Según el términos de regularización empleado, se modificará de una u otra manera la solución del problema de optimización.

La norma L1 se define como:

$$\sum_{j=1}^{n} |u_j| \quad (4)$$

donde u_j son las componentes del vector u sobre el que se aplica el términos de regularización.

Con esta norma se describe la regresión Lasso, la cual produce modelos más fáciles de interpretar, ya que Lasso favorece la aparición de un mayor número de valores nulos en la solución. Está limitado cuando el número de muestras es relativamente pequeño comparado con la dimensión de éstas.

La norma L2 se define como:

$$\sum_{j=1}^{n} |u_j|^2 \quad (5)$$

Esta norma se aplica en las regresiones Ridge con el fin de contraer los coeficientes estimados pero sin llegar a cero, por lo que no se consigue la selección de variables. Este tipo de regresión se propuso con el fin de minimizar el problema de la colinealidad que se define cuando el valor de correlación entre algunas de las variables es cercano a la unidad.

Finalmente, el término de regularización de Elastic Net se define como:

$$\gamma_1|u_1| + \gamma_2|u_2|^2 \quad (6)$$

La cual se puede reescribir de la forma definida en MLlib:

$$\alpha|u_1| + (1-\alpha)|u_2|^2 \quad ; \text{ siendo } \alpha = \frac{\gamma_2}{\gamma_2\gamma_1} \quad (7)$$

Esta penalización es una combinación convexa de las dos anteriores. Como puede comprobarse eligiendo el valor de α igual a 1 es equivalente al modelo Lasso y eligiendo α igual a 0 al modelo Ridge. El término cuadrático de este término resuelve la principal limitación de Lasso, que se da cuando nos encontramos con datos con dimensiones altas y contamos con pocas observaciones y fomenta el efecto de grupo. La parte correspondiente a L1 genera un modelo disperso.

En la Figura 4 podemos ver la geometría de los términos de regularización presentados en esta sección, usando $\alpha = 0.5$ para Elastic Net.

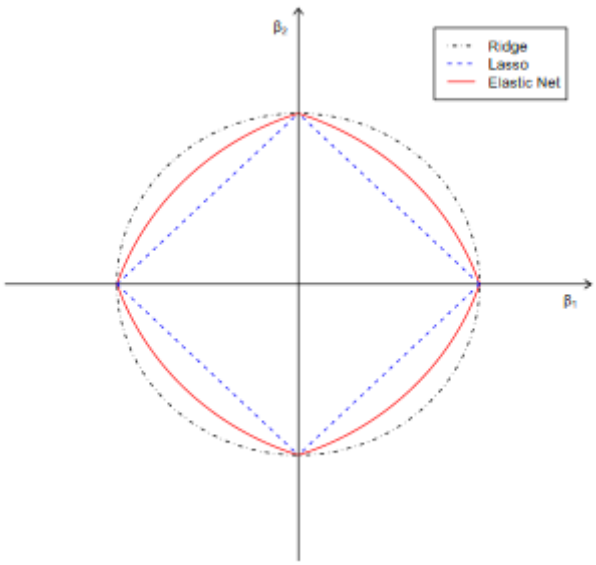


Figura 4. Geometría de términos de regularización. Tomada de [15]

Estas regularizaciones son algunas de las posibilidades más inmediatas que se pueden aplicar para el término $R(U)$ en la ecuación (1). La elección del parámetro γ es también importante ya que permite compensar la importancia del término de regularización: cuanto menor es su valor, menor es la penalización. Para la elección de este parámetro se suelen contemplar dos soluciones. La primera de ellas consiste en utilizar una traza de Ridge, representar los coeficientes de regresión en función de γ, y elegir el que los estabiliza. Otra de las soluciones consiste en estimar γ mediante validación cruzada[3].

3.3 Tipos de normalización

Con la normalización lo que se pretende es ajustar todas las variables a una misma distribución con la misma media y varianza, ajustarla dentro de un rango, en percentiles etc. Cuando los datos están normalizados se consigue un comportamiento numéricamente más estable de los métodos de optimización.

Scikit-Learn proporciona varios tipos de normalización, pero ya que deseamos que dicha normalización se aplique sobre datos distribuidos en Spark, emplearemos las funciones que proporciona MLlib. Dicha librería permite normalizar los datos aplicando una transformación lineal que los escala y desplaza para que tengan media cero y varianza unidad. En cualquier caso, nuestra librería para MVA permite al usuario seleccionar la aplicación o no de la normalización, calculando la transformación a aplicar a partir de los datos de entrenamiento.

[3] Validación cruzada: es un método que consiste en la partición aleatoria de los datos en k subconjuntos (uno de prueba y k-1 de entrenamiento) iterando sobre los subconjuntos y aplicando el proceso de validación cruzada, finalmente se aplica la media aritmética de todos los resultados para estimar la capacidad de generalización del modelo para los ajustes empleados en cada caso.

3.4. Descripción de la toolbox MVA

Para la descripción completa de la implementación de la clase de la toolbox procederé a realizar dos tipos de desarrollo. Por un lado, una explicación de la estructura del software implementado, detallando en cada función que compone la toolbox: su funcionalidad general, los tipos de datos que acepta y las funciones de librerías que se usan. Por otro lado, realizaré una explicación funcional de cómo se desarrollaría la puesta en marcha de la toolbox.

3.4.1. Formato de datos de entrada aceptados

Las rutinas implementadas permiten fundamentalmente aplicar técnicas MVA cuando los datos objetivo (espacio de salida) constituyen los valores deseados de un problema de regresión o la etiqueta de la clase a la que pertenece el dato en problemas de clasificación. Esto conlleva que la estructura que deben de tener los datos de entrenamiento tiene que seguir ciertas especificaciones.

Un tipo de dato muy utilizado en aplicaciones de MLLib es el *Labeled point* [16]. Este tipo consta de un vector de entrada que puede ser *denso* o *disperso* y la etiqueta respuesta correspondiente, *LabeledPoint(* etiqueta, vector). Se reconoce como vector *dense* a los arrays de NumPy y a las listas de Python. Los vectores *sparse* se pueden definir con la librería de vectores de MLlib y con la librería de Python SciPy, aunque la más común es la primera. Un vector normal sería (5.0, 0.0, 1.0, 3.0) que en formato *sparse* quedaría representado como (4, [0, 2, 3] , [5.0, 1.0, 3.0]). En los casos en que los datos presentan gran cantidad de ceros, la representación sparse es más eficiente en términos de memoria y conduce habitualmente a implementaciones más eficientes [17]. Por lo tanto, para problemas de clasificación uno de los tipos de datos que acepta es un RDD de *LabeledPoint*.

Otro de los datos que acepta la clase es un RDD de tuplas compuestas por NumPy arrays. Para un problema de regresión la tupla esperada sería ($X = [x1, x2, \dots, x_n]$, $Y = [y1, y2, \dots, y_m]$) y para un problema de clasificación ($X = [x1, x2, \dots, x_n]$, $[0\ 0\ 0\ 0\ 1\ 0\ \dots]$) donde el segundo elemento de esta tupla es un vector binario con un uno en la posición correspondiente a la clase.

Por último, el otro tipo de datos de entrada que contempla la clase se da cuando el tipo de MVA elegido es PCA. Ya que para este tipo X=Y por lo tanto se espera un RDD de NumPy arrays.

3.4.2. Descripción estructural

Las funciones de las que consta la clase son las siguientes:

- La primera función de la clase MVA es *__init__().* Esta función es invocada automáticamente cuando se instancia la clase y como argumentos requiere:
    - El tipo de método MVA (*typeMVA*) que se va a utilizar. Los valores que acepta son 'PCA', 'CCA' y 'OPLS'.
    - El tipo de regularización (*typeReg).* Los valores que acepta son: 'l1' para la norma L1 (este valor también se tomará por defecto) y 'l2' para la norma L2.
    - El tipo de normalización (*typeNorm).* Los valores que acepta son es *'norm'*, si se quiere normalizar o '*None'* si no.
    - Un valor para la condición de parada (*tol).* Este valor se emplea para frenar la ejecución de la clase cuando la norma de Frobenius entre la matriz U de la ejecución

anterior y la matriz U la ejecución en proceso sea menor que este valor, lo que significa que el algoritmo ha convergido.

- El número de características que queremos extraer (*numFeatures).*
- El parámetro de regularización (*regParam)* por defecto toma el valor de 0.01.
- El tamaño de paso que se usa en cada iteración de SGD (*step)* del paso U*;* por defecto toma el valor de 1e-3.
- El número máximo de iteraciones para el SGD del paso U (*iterations);* por defecto toma el valor de 100
- El número máximo de pasos *U* (*max_Ustep)* que se permiten para que el algoritmo converja; su valor por defecto es 10. Si se supera este número máximo de pasos, la optimización concluye y se devuelve la matriz de proyección obtenida en el último paso del algoritmo.

- La función *prepareData()* valida los datos de entrada proporcionados por el usuario, y los convierte al formato necesario para el uso interno de la toolbox. Si recibe un *LabeledPoint,* binariza la etiqueta por medio de la función *label_binarize()* de la librería de Sckit-Learn y transforma el vector en un *DenseVector. S*i el método seleccionado es PCA, sólo nos quedamos con el array de características y lo transformamos en una tupla (X , X) de vectores *dense*. Además, se comprueba que los tamaños de todos los elementos del RDD sean compatibles.

- La función *calcCov()* calcula las dos matrices de covarianzas que se van a necesitar en diferentes puntos de la clase: la matriz de covarianzas entre Y y la matriz entre Y y X. El argumento requerido para esta función es el tipo de matriz a calcular (*typeCov),* que puede tomar valores de *'Cyx'* o *'Cyy'.*

- La función *createOmega()* crea la matriz Omega definida anteriormente para los diferentes tipos de MVA.

- La función *calcFrobeniusNorm().* Esta función calcula la norma de Frobenius entre dos matrices. Como parámetros de entrada tiene las dos matrices necesarias para el cálculo.

- La función *normalizer()* normaliza los datos de entrenamiento. Dependiendo de si el usuario quiere o no normalizar se lleva a cabo un procedimiento u otro. Si quiere normalizar se hace uso de la función de MLlib *StandarScaler()* la cual elimina la media y escala a uno la desviación típica de los datos de entrenamiento. Para el vector de características hacemos la transformación con la media y la desviación típica, mientras que para el vector de etiquetas sólo eliminamos la media. Si el usuario no quiere normalizar solamente se elimina la media al vector de características por el mismo procedimiento de antes, pero se ignora el reescalado según la desviación típica. Para finalizar, se crea un nuevo RDD de LabeledPoint con los datos normalizados.

- *stepU()*: esta función realiza el paso U explicado previamente, resolviendo el problema de regresión mediante la función de MLlib *LinearRegressionWithSGD().* Con esta función se entrena el modelo de regresión usando Stochastic Gradient Descent (SGD) el cual resuelve la fórmula: $f(W) = \frac{1}{2n} |XW - Y|^2$ correspondiente al MSE, o la función modificada mediante la inclusión del término de regularización correspondiente.

  Los parámetros de entrada que recibe la función *stepU*() son las matrices *W* y *Omega, el parámetro R* que corresponde al número de proyecciones a extraer, y el parámetro de regularización (*regParam)* . Como salida obtendremos la matriz *U* con dimensiones *R x M*. La matriz

U de salida se va rellenando con los pesos generados por el proceso iterativo de regresión en el rango de *R.*

- *stepW(),* esta función resuelve el paso W descrito anteriormente como un problema de autovalores con la función *linalg.svd()* de la librería NumPy además de varios cálculos previos con matrices. Como parámetros de entrada requiere la matriz U, la matriz de covarianzas entre Y y X (*Cyx*), la matriz Omega y su inversa.

- La función *computeMSE()* calcula el mínimo error cuadrático entre los datos objetivo y sus reconstrucciones basadas en las matrices U y W. Los parámetros de entrada que necesita son: la matriz U, la matriz W y los datos de entrenamiento (*trainingData)*.

- La función *fit()* ajusta el modelo y devuelve la matriz U con los coeficientes correspondientes. Como parámetros de entrada solo requiere los datos a entrenar. Esta función sigue los siguientes pasos: en primer lugar prepara los datos mediante la función *prepareData(),* lo siguiente que hace es crear la matriz omega correspondiente con *createOmega()* y normaliza los datos de entrada mediante *normalizer().* Lo siguiente que hace es empezar el proceso iterativo por el cual va haciendo múltiples cálculos de las matrices U y W (iterando los *stepU y stepW*). Las iteraciones paran según la condición de parada *tol*.

- Por último la función *predict()* devuelve el RDD con las características extraídas. Como parámetro de entrada requiere los datos a procesar. Lo primero que hace es normalizar los datos de entrada con los mismos valores para la media y la varianza que resultan de aplicar la función *normalizer()* en la parte de ajuste del modelo. Realiza la operación $X^{NEW} = U^T * X$ para extraer las características relevantes. Es importante

haber ajustado el modelo previamente y como argumento de entrada necesita el RDD de datos.

3.4.2 Descripción funcional

Para una ejecución normal de la herramienta lo primero que necesitamos es cargar los datos y paralelizarlos en un RDD. Una vez que los tenemos, creamos un objeto de clase MVA indicando obligatoriamente: el tipo de MVA que queremos usar, el tipo de regularización, si queremos normalizar los datos o no, un valor para la condición de parada y el número de variables que queremos estudiar. Como valores opcionales, si no se tomaran los valores por defecto, han de especificarse un valor para el parámetro de regularización, seguido de un valor para el paso, el número máximo de iteraciones y un número máximo de pasos *U*. Eligiendo PCA, regularización L1, sin normalizar, 1e-5 de tolerancia, 7 características a estudiar, 0.0001 como parámetro de regularización, un valor de 1e-3 para el paso y 100 iteraciones para el algoritmo obtenemos una instrucción de la forma:

```
#Creación del objeto MVA
prueba = MVA('PCA','l1','None',1e-5,7,0.0001,1e-3,100)
```

Figura 5. Creación del objeto MVA

Después de esta instrucción el objeto MVA se habrá creado y las variables privadas de la clase estarán accesibles. El siguiente paso consistirá en ajustar el modelo llamando a la función *fit()*. Esta función como se explicó en el apartado anterior prepara los datos y realiza los procesos iterativos *stepU()* y *stepW()* con el fin de generar la matriz U. La llamada a la función *fit()* sería:

```
prueba.fit(RDD_PCA)
```

Figura 6. Ejecución de la función fit()

Por último llamamos a la función *predict()* donde se transformarán los datos de entrada en unos nuevos que contienen las características más relevantes. La llamada a esta función será:

```
RDD_new=prueba.predict(RDD_PCA)
```

Figura 7. Ejecución de la función predict()

Opcionalmente también podemos utilizar la función *computeMSE()* para calcular el error que se ha cometido al ajustar el modelo.

Capítulo 4. Evaluación

En el presente capítulo se va a realizar la validación del software implementado con dos tipos de datos diferentes: sintéticos y una base de datos real. Se procederá a realizar diferentes experimentos que acrediten sus funcionalidades para los casos de PCA y OPLS, ya que CCA puede verse como una modificación de OPLS para eliminar su dependencia con las varianzas de las variables objetivo. Por otro lado, también se evaluará la escalabilidad de la solución.

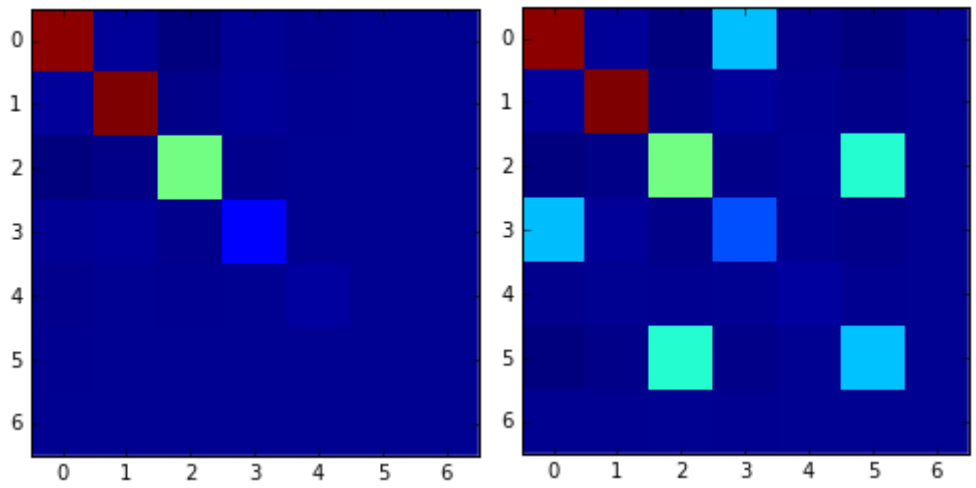
4.1 Experimentos con Datos sintéticos

La primera evaluación se va a realizar sobre datos generados artificialmente. Dichos datos han sido diseñados para ilustrar determinadas propiedades de los métodos considerados. Los datos se generan a partir de matriz aleatoria, Z, con 5000 patrones, cada uno con 7 variables. Dichas variables se han generado a partir de muestras tomadas de variables gaussianas con medias nulas, y varianzas $1, 1, 0.5,\ 0.1,\ 1e-2, 1e-8,$ y $1e-8$; con esto se pretende dar relevancia a las 4 primeras variables confiriéndoles una mayor varianza frente a las 3 últimas. A partir de dicha matriz Z, se han generado las representaciones de entrada y salida de la siguiente manera:

- Para la obtención de la salida, Y, se ha creado una matriz de coeficientes de regresión de dimensiones 7 por 3 donde se ha dado valores más altos a la quinta variable, es decir mayor relevancia frente a las demás. De esta manera, esperamos que OPLS sea capaz de detectar la importancia de dicha variable, a diferencia de los métodos no supervisados que la descartarán por su baja varianza.

La matriz de entrada X se ha generado promediando alguna de las variables originales en la matriz Z. En la Figura 8 puede verse el resultado

de la matriz de covarianzas, en (a) se muestra la covarianza de las variables de Z, mientras que en (b) se muestra la covarianza de las variables de X, donde puede comprobarse que la variable 0 está correlacionada con la 3 y la 2 con la 5.



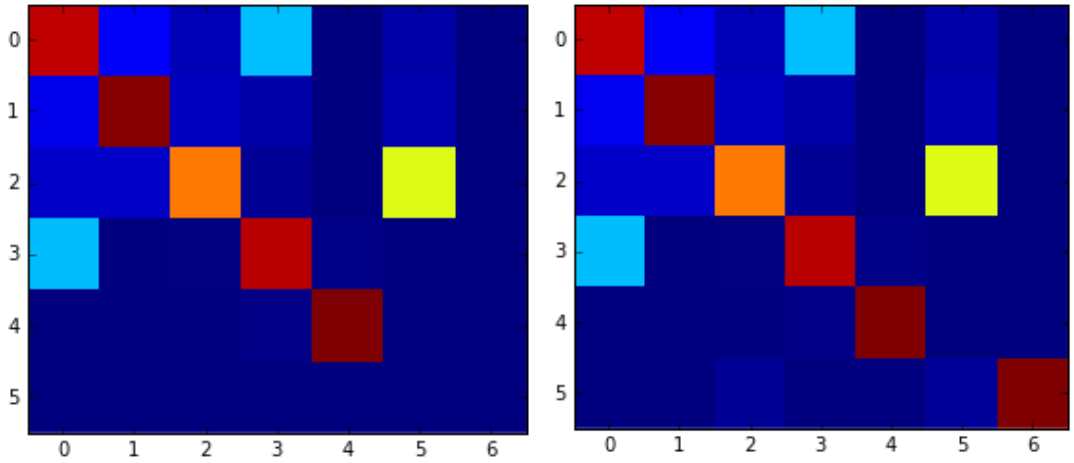
(a) (b)

Figura 8. Matriz de covarianzas

4.1.1 Comprobación de PCA y OPLS en ausencia de regularización

La primera prueba que se va a realizar es comprobar que la solución obtenida por la toolbox para el método PCA es equivalente a la obtenida por la implementación de Scikit-Learn, a partir de ahora PCA_SL. Para ello se entrena el modelo de la toolbox con PCA, sin normalizar y con un valor de parámetro de regularización virtualmente nulo, ya que PCA_SL no contempla la implementación de soluciones regularizadas. Por otro lado, se entrena PCA_SL con un número de componentes igual a 6. En la Figura 9 se ha representado en (a) la matriz U obtenidas por la implementación de la toolbox y en (b) la matriz de componentes principales obtenidas por PCA_SL. Como puede

comprobarse, el resultado es exactamente el mismo para las 5 primeras componentes. Únicamente se observa una diferencia en la sexta componente, donde el PCA de la toolbox no parece identificar ninguna componente con varianza significativa. En cualquier caso, la coincidencia de las componentes asociadas a variables con varianza significativa da muestra de la correcta convergencia de nuestra implementación a la solución deseada.



(a) (b)

Figura 9. Resultados de PCA

Comprobamos ahora la capacidad de OPLS para identificar la variable más correlacionada con los datos de salida. Vemos en la Figura 10 que el autovalor más grande se asigna a un vector de proyección que selecciona de forma casi exclusiva la quinta variable. Este es el comportamiento deseado ya que, aunque las tres primeras variables tenga una varianza mayor, OPLS selecciona las características que permiten aproximar mejor la matriz de datos de salida, Y.

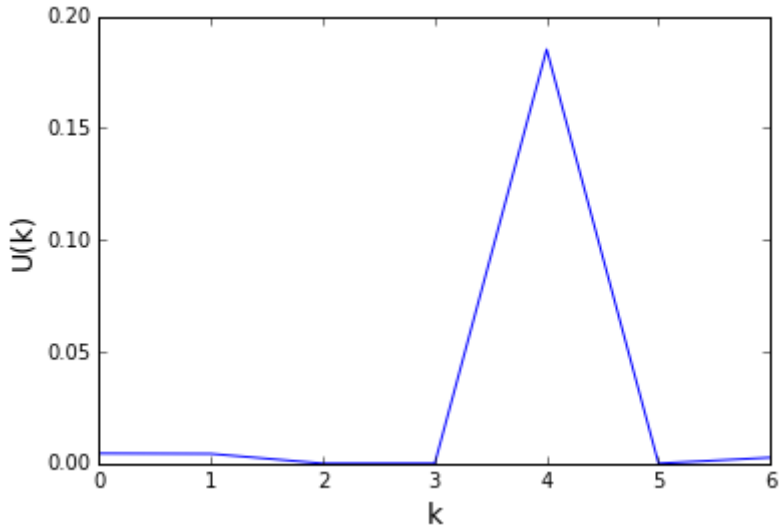


Figura 10. Proyección de OPLS

4.1.2 Efectos de incremento de la regularización

En este apartado el objetivo es visualizar cómo con un aumento de la constante de regularización obtenemos soluciones más dispersas, es decir obtenemos coeficientes de las matriz U más pequeños y como consecuencia más ceros. Para ello se ha entrenado el modelo con tres valores de regularización diferentes: 0.0001, 0.01 y 0.1. En la Figura 11 se han representado cada uno de los 6 vectores de proyección para las distintas constantes de regularización.

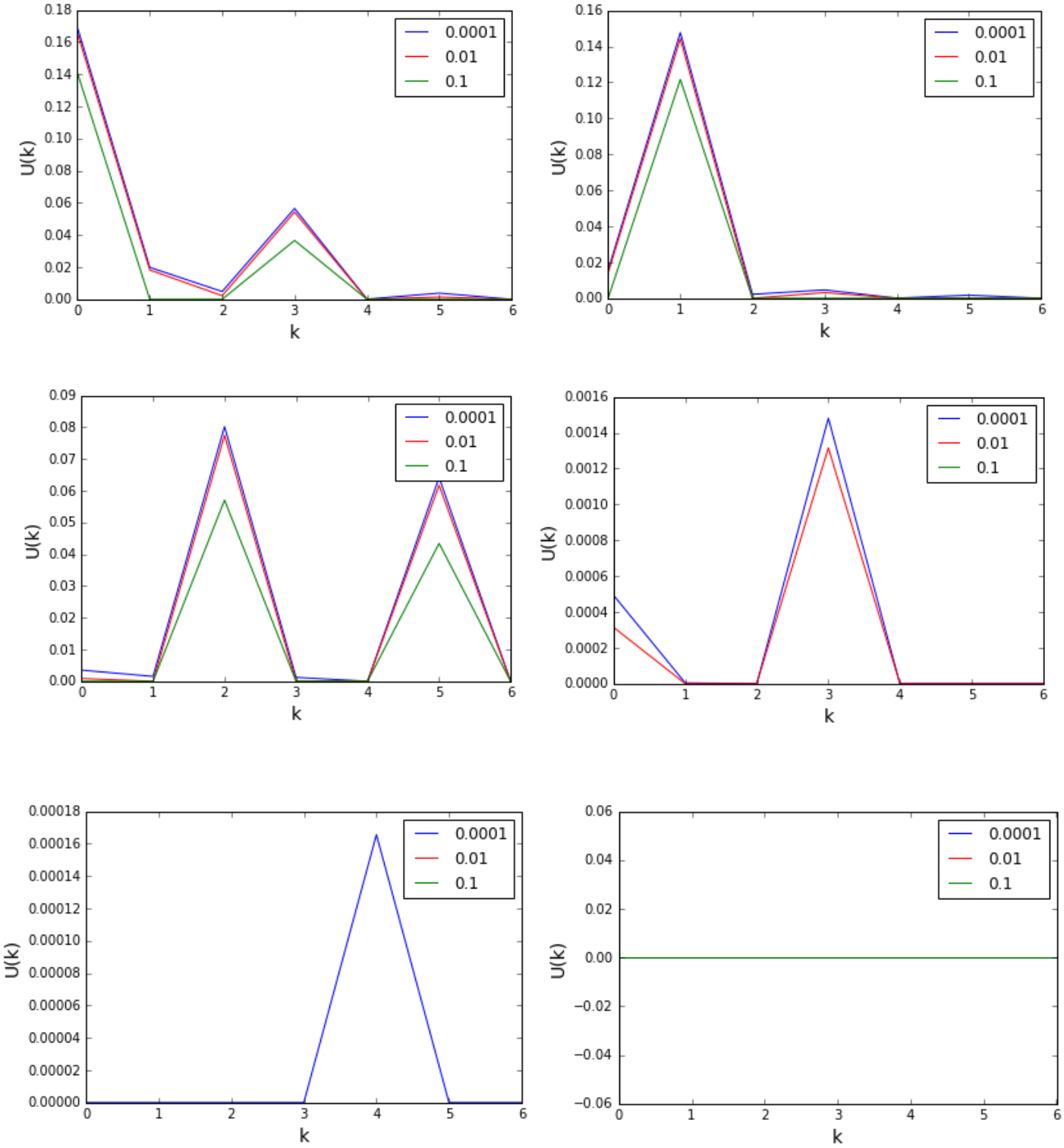


Figura 11. Vectores de proyección

4.1.3 Blanqueado de datos de entrada

El siguiente efecto que se va a estudiar es cómo se logra el blanqueo de los datos proyectados, i.e., $X_2 = XU^T$, donde X_2 hace referencia a los datos transformados, según variamos la regularización. Tal y como se explica en [9], la implementación considerada en este proyecto trata de forzar la incorrelación de las distintas variables extraídas de los datos. De esta manera, conseguimos que la mayor relevancia sea para la primera de las variables, la segunda para la segunda y así sucesivamente. Además, la incorrelación garantiza que si vamos aumentado el número de variables seleccionadas vamos añadiendo nueva información (no repetida) con cada variable.

Para esta comprobación se va a estudiar la matriz de covarianzas de X_2 sumando todos sus elementos (en valor absoluto) menos aquellos que pertenecen a la diagonal principal, con el propósito de verificar cómo los datos están menos blanqueados a medida que se aumenta la regularización. Se ha entrenado el modelo con valores de regularización correspondientes a 0.00001, 0.001, 0.1 y 0.5 obteniendo valores de la suma de sus elementos los cuales pueden apreciarse en la Figura 12.

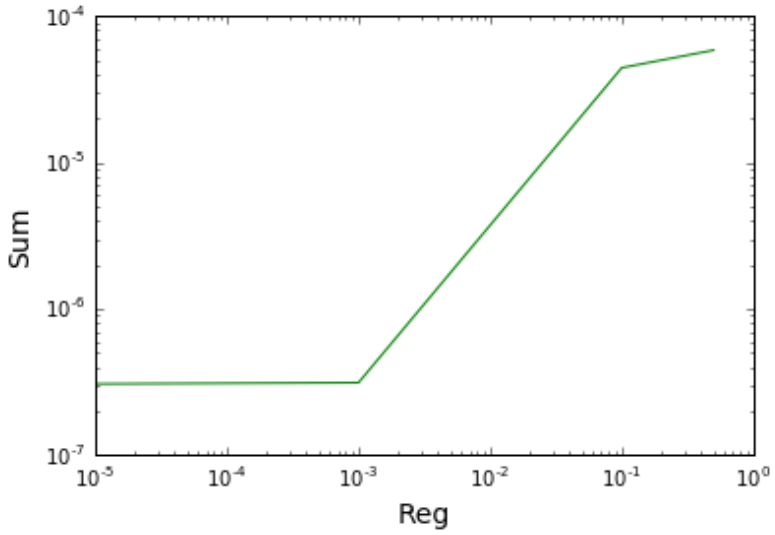


Figura 12. Suma de los elementos de la matriz en función de la regularización

4.2 Base de datos real

La base de datos[4] real que ha sido utilizada consiste en un total de 4435 muestras cada una de ellas correspondiente a un pixel de una imagen tomada por satélite. La finalidad de esta base de datos es un problema de clasificación por lo que cada muestra consta de 36 atributos referidos a los valores de los píxeles en diferentes bandas espectrales de la imagen con valores de 0 a 255 y una etiqueta, con valores entre 1 y 7, identificando el tipo de terreno asociado a cada píxel de la imagen.

4.2.1 Escalabilidad de los datos

La escalabilidad de un método de aprendizaje automático es una de las propiedades más importantes, sobre todo en aplicaciones de big data. Por ejemplo, un método cuya complejidad crece cuadráticamente con el número de datos de entrenamiento puede resultar inabordable para grandes bases de datos. En nuestro caso, es de esperar un crecimiento lineal de la complejidad, ya que el cálculo de las matrices de covarianza incrementa de esa manera con el número de datos, y la resolución de los problemas de regresión y descomposición en autovalores tienen una complejidad prácticamente constante con independencia del número de datos de entrenatmiento.

Para la demostración de este punto se ha procedido a hacer tres particiones de las muestras con el fin de calcular el tiempo que emplea en ejecutar el entrenamiento del modelo para PCA para cada una de las particiones. Las particiones que se han realizado son: del 1% que

[4] URL de la base de datos. https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/satimage/

corresponden a 37 muestras, 10% correspondiente a 410 muestras y finalmente para el 100% 4435 muestras. El número de muestras resultante no coincide exactamente con el porcentaje correspondiente debido a la función sample() empleada, ya que hace un filter() en el que cada dato pasa o no según el resultado de un experimento aleatorio con una probabilidad dada. En la Figura 13 puede comprobarse de hecho que el tiempo de ejecución crece con el logaritmo del número de muestras de entrenamiento. Convendría realizar experimentos con bases de datos mayores para poder concluir de manera más exacta cuál es la escalabilidad de la toolbox con el tamaño de los datos de entrenamiento y su dimensionalidad.

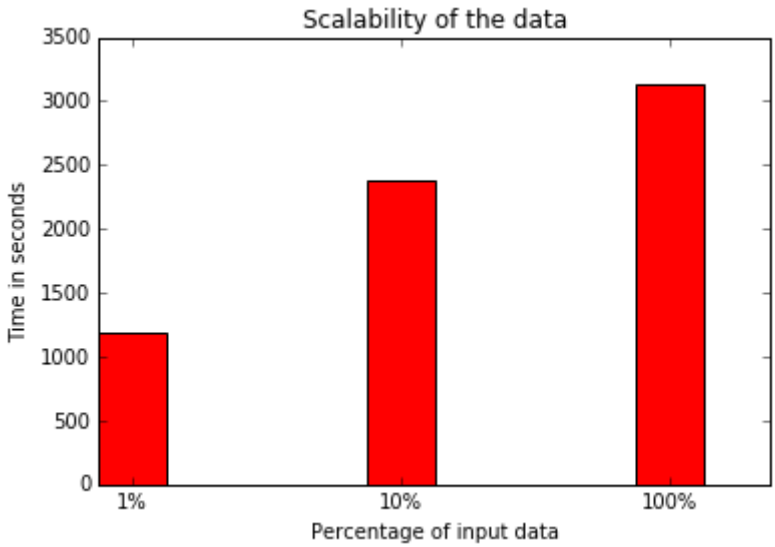


Figura 13. Escalabilidad de los datos de entrada

4.2.2 Función objetivo según el número de proyecciones

Con la función objetivo se hace referencia al cálculo del mínimo error cuadrático (MSE) que se obtiene al ir seleccionando un número mayor de características extraídas. Para el caso de los datos de entrada se ha ido entrenando el modelo variando el número de variables extraídas de 1 a 25 y evaluando la función: $|X - WU^{T}X|^{2}$ para PCA. Como se puede comprobar en la Figura 14, el valor de MSE disminuye a medida que se van extrayendo más características.

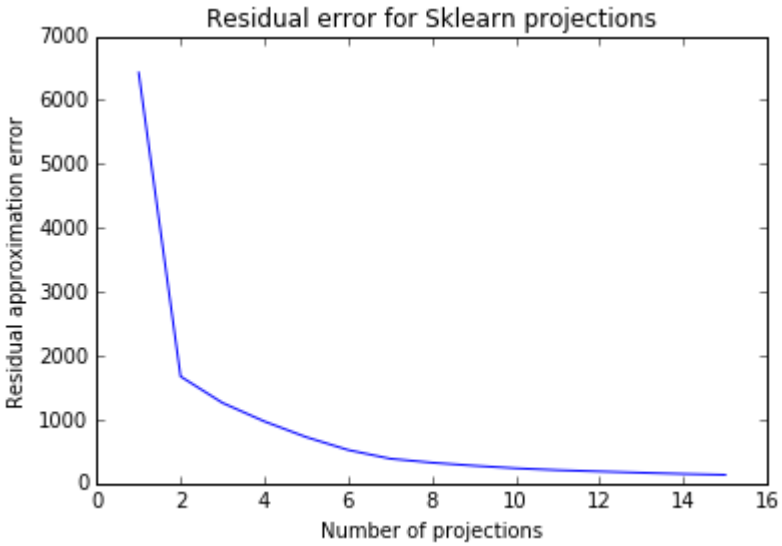


Figura 14. MSE en función del número de características extraídas

Capítulo 5. Planificación del trabajo y presupuesto detallado.

En este capítulo se explicará cómo se ha desarrollado el proyecto en el tiempo y el presupuesto estimado que requeriría.

5.1. Planificación del trabajo

El desarrollo que ha llevado el proyecto ha contado con varios altibajos en su desarrollo ya que no ha tenido una dedicación regular. El desarrollo puede resumirse en unas seis actividades:

1. Aprendizaje del lenguaje Python por medio de la plataforma Code Academy[5].

2. Realización del MOOC “Big Data Analysis with Apache Spark” de la plataforma edX para el aprendizaje de Apache Spark y de sus herramientas básicas[6].

3. Comprensión del código proporcionado por el departamento relacionado con la solución.

4. Implementación del código de la toolbox .

[5] https://www.codecademy.com/learn/python
[6] https://www.edx.org/course/big-data-analysis-apache-spark-uc-berkeleyx-cs110x

5. Realización de pruebas de la API con una base de datos real y datos sintéticos.

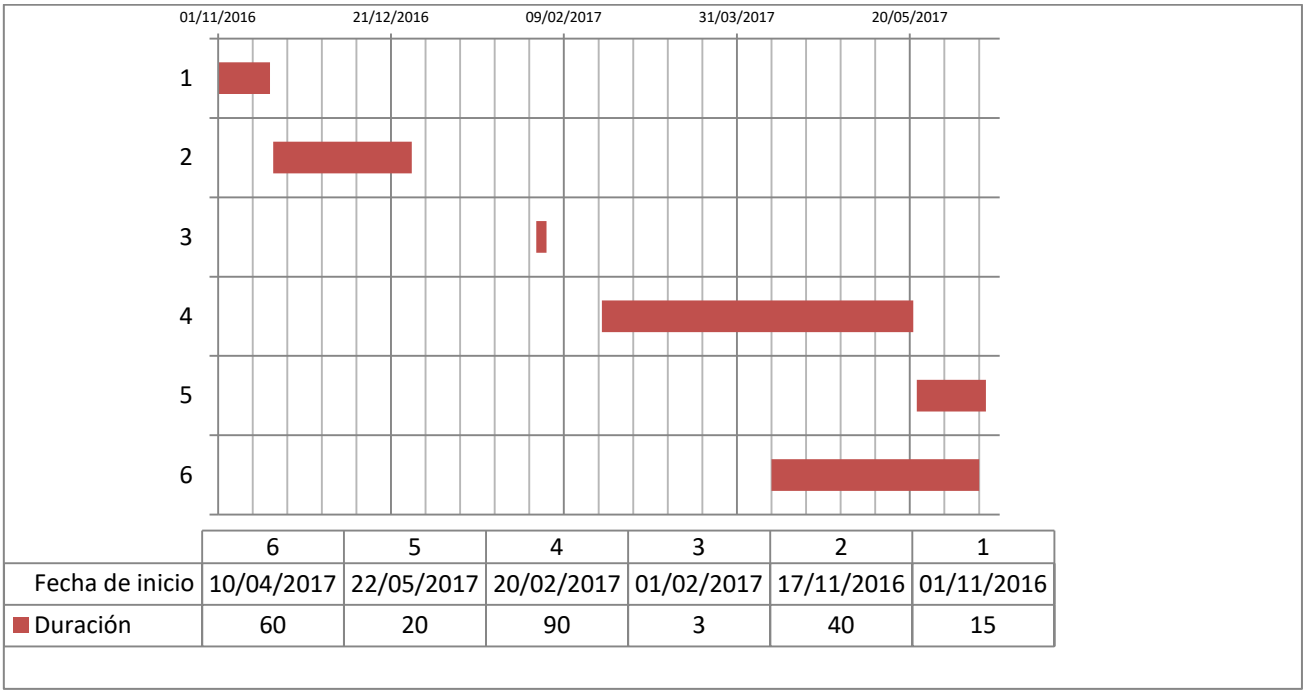
6. Escritura de la memoria del proyecto.

En la Tabla 3 se detalla la duración de cada actividad con sus fechas de inicio y final:

Actividad	Fecha de Inicio	Duración (días)	Fecha de terminación
1	01/11/2016	15	15/11/2016
2	17/11/2016	40	28/12/2016
3	01/02/2017	3	03/02/2017
4	20/02/2017	90	20/04/2017
5	22/05/2017	20	10/06/2017
6	10/04/2017	60	15/06/2017

Tabla 3: Duración de las actividades

En la Figura 15 se puede observar el diagrama de Gantt con el objetivo de representar gráficamente el desarrollo que ha tenido cada actividad:



	6	5	4	3	2	1
Fecha de inicio	10/04/2017	22/05/2017	20/02/2017	01/02/2017	17/11/2016	01/11/2016
Duración	60	20	90	3	40	15

Figura 15. Diagrama de Gantt

5.2 Presupuesto

Para la estimación del presupuesto se han tenido en cuenta los costes de personal y los costes de software y hardware utilizados.

En la Tabla 4 se detallan los costes de personal, considerando una media de 2 horas de trabajo por día de dedicación, con un coste 10 euros la hora.

Nombre	Categoría	Coste hora	Horas totales	Coste total
Alba García-Tembleque	Ingeniero Junior	10	456	4560 €
Total				4560 €

Tabla 4: Coste de personal

En la Tabla 5 se especifica los recursos software y hardware utilizados con el coste de cada uno de ellos y el coste que ha supuesto para el proyecto teniendo en cuenta la amortización[7] y la duración.

Recurso	Precio(€)	Amortización(%)	Duración(Meses)	Coste (€)
Ordenador portátil Acer	540	26	8	93.6
Databricks	Gratuito	-	6	0
Nodo de computación	4.500	26	1	97.5
Total				191.1

Tabla 5: Recursos utilizados y su coste

Por lo tanto, el coste total especificado para este proyecto, teniendo en cuenta los gastos de personal y de amortización de recursos materiales, asciende a la cantidad reflejada en la Tabla 6.

Descripción	Precio
Coste de personal	4560€
Coste de recursos	191.1€
Total	4751.1€

Tabla 6: Coste total del proyecto

[7] La amortización que se ha tenido en cuenta es la correspondiente a un año.

Capítulo 6. Impacto socioeconómico

En este capítulo se analizará el impacto económico y social que se esperaría después de la implantación del proyecto.

6.1 Impacto social

El impacto social de este proyecto tiene como base el impacto que supone para la sociedad digital actual el Software Libre. Este punto está estrechamente relacionado con el siguiente capítulo dónde se explicará más detalladamente este concepto.

La tecnología generada en código abierto es útil para otros usuarios, ya que permite continuar generando conocimiento realizando modificaciones sobre ella, compartiendo los datos generados etc. Esta visión de apertura ha permitido que hoy en día exista gran cantidad de software desarrollado y puesto a disposición del público en general bajo diversas modalidades de distribución de código en abierto, siendo dicha actividad muy relevante en el contexto del Big Data que es el tema que nos ocupa.

En los últimos años, grandes empresas se están beneficiando del potencial de los grandes volúmenes de datos y por consiguiente de plataformas de código abierto que lo manipulan como es el ejemplo de Scikit Learn, así como de otras aplicaciones finales. Estos proyectos crean valor y datos y asientan las bases de una nueva forma de generar conocimiento, expandiéndolo.

Las APIs (*Application Programming Interface)* son una de las grandes contribuidoras a este nuevo modelo contando con ejemplos cotidianos como son la API de Twitter, todas las de Google, etc. Las APIs permiten que otras aplicaciones hagan uso de ellas para temas que pueden tener relación con otros muy diferentes aportando así nuevas ideas y usos [18]. Es importante remarcar el hecho de que si además una API está disponible en repositorios abiertos, facilita mucho más su uso y propicia la creación de productos colaborativos. Cabe destacar también que no solo se ha diseñado una API como tal ya que también se han programado las funciones que la componen, de ahí la definición de toolbox, pero para el análisis del impacto pueden considerarse como análogas.

En concreto, con la toolbox MVA creada se pretende proveer al público en general de una herramienta eficaz de extracción de características con la cual poder reducir la dimensión de los datos. Es fácil ver las posibilidades de integración en otras aplicaciones que requieran filtrado para conjuntos de datos con muchas mediciones.

Esta toolbox ha sido creada con el deseo de accesibilidad y de pertenencia a un proyecto común de colaboración que permita a otros usuarios crear productos derivados y distribuirlos y sacar valor de ellos, lo que justifica su publicación por medio de un repositorio abierto de Github.

6.2 Impacto económico

El impacto económico de este proyecto es difícil estimarlo por sí sólo. Sin embargo, ya que está enfocado para ser parte de proyectos más grandes en el entorno del Big Data y Aprendizaje Automático, sí que puede medirse el

impacto económico que tienen estos dos términos y referenciarlo a nuestro proyecto en cuestión.

En los próximos años dentro de las profesiones más demandadas estará la de especialista en análisis de datos, estimándose que en Europa harán falta 1,3 millones de estos profesionales para satisfacer la demanda prevista [19]. Las empresas, públicas y privadas, han cambiado su mentalidad percatándose de la importancia que tiene una buena interpretación de los datos. El Big Data, aparte de crear valor con nuevos negocios, ahorra muchos costes en la administración.

Un gran número de sectores públicos se han beneficiado ya de las tecnologías Big Data, como son los casos del sector salud donde el anticiparse a la enfermedad es lo primordial siendo capaces de medir la actividad física, o las calorías ingeridas con el fin de conocer más al paciente y registrar toda su información. El sector turismo, ha empezado a hacer un gran uso dejando a un lado las tradicionales encuestas y se evalúan las páginas web visitadas, mensajes en redes sociales, ubicaciones de GPS todo para ofrecer recomendaciones personales y en tiempo real.

Para el año 2020 se estima que el impacto del desarrollo del Big Data en el PIB europeo sea de un 1,9%, en el comercio un 23%, industria 22%, en la administración 13% y en el sector financiero un 13% [19].

Capítulo 7. Marco regulador

En el presente capítulo se analizará el marco regulador en el que está enmarcado el software libre. Se explicarán las licencias de las que se puede hacer uso y se elegirá una para la Toolbox implementada en este Trabajo de Fin de Grado.

7.1. Software libre y Open Source

El código de la toolbox en cuestión estará disponible en la plataforma de desarrollo colaborativo de software GitHub. Esta plataforma permite compartir proyectos de desarrollo software, y por supuesto usar repositorios ajenos clonándolos en tu propia cuenta con el fin de trabajar de manera colaborativa. Ofrece además muchas herramientas interesantes para el trabajo en equipo.

Github trabaja con el software de control de versiones Git. Git es un sistema distribuido de control de código fuente que permite ver la evolución de nuestro proyecto, recuperar versiones anteriores, analizar la actividad de los colaboradores, etc. El uso de estas plataformas se ha incrementado notablemente no sólo por su eficacia y sencillez si no por ser comunidades Open Source.

El término Open Source (Código abierto) hace referencia al software distribuido y desarrollado libremente donde se comparte el código fuente y puede ser modificado previa autorización. La máxima de este concepto recae en que el usuario haga uso del código, lo desarrolle e incluso mejore para

adaptarlo a sus propias necesidades [20]. Cabe resaltar la diferencia entre los movimientos Open Source y Software Libre.

El Software Libre está enfocado desde un punto de vista más filosófico y ético. La organización Free Software Foundation (FSF) establece que las libertades que tiene que respetar un Software Libre son: libertad de uso de código para cualquier propósito, libertad de estudio, modificación y hacer públicas esas mejoras y libertad de distribución de las mismas [21].

La clara diferencia radica en que el movimiento del Software Libre hace hincapié en cuestiones éticas, mientas que para el software Open Source es más importante el beneficio obtenido a gracias a las mejoras que pueda sufrir el software. Es importante destacar también que los trabajos derivados de un código abierto o de un Software Libre no tienen porque compartir las mismas licencias, esto es, que no tiene porque ser también un Software Libre o Open Source.

Estos dos principios reconocen el mismo conjunto de licencias que cada vez más estas plataformas animan a incluir en los repositorios alojados en sus servidores.

7.2 Licencias

El estudio de las licencias de las que hace uso el Software Libre y Open Source se va a centrar en aquellas que más se usan en los proyectos software en

GitHub, ya que ésta es la plataforma utilizada. En la Figura 16 podemos ver las licencias más usadas y el número de proyectos en porcentaje que las utilizan:

Rank	License	% of projects
1	MIT	44.69%
2	Other	15.68%
3	GPLv2	12.96%
4	Apache	11.19%
5	GPLv3	8.88%
6	BSD 3-clause	4.53%
7	Unlicense	1.87%
8	BSD 2-clause	1.70%
9	LGPLv3	1.30%
10	AGPLv3	1.05%

Figura 16. Ranking de licencias más populares en GitHub. Tomada de [22]

La licencia MIT permite el uso comercial del proyecto, su distribución, modificación y uso privado con la condición de proveer de la contribución hecha y de ser avisado. Las modificaciones y trabajos futuros tienen que llevar la licencia y el copyright propio. Esta licencia incluye limitación de la responsabilidad y explícitamente indica que el proyecto no ofrece ninguna garantía.

Por otro lado, la licencia Apache cuenta con permisos de uso comercial, distribución, modificación, uso de los derechos de patente para contribuidores y uso privado. Se debe incluir una copia de la licencia y de los derechos de copyright, y los cambios efectuados en el código deben ser documentados. Entre las limitaciones que incluye destacan la de responsabilidad, la no oferta de garantía, y la no concesión de derechos de marcas registradas.

La licencia GPLv3 (GNU General Public License v3.0) comparte los mismos permisos que las anteriores, incluyendo la expresa concesión de los derechos de patente para los contribuidores. Añade la condición de que el código fuente debe estar disponible cuando el software se distribuya, exigiéndose además la inclusión de una copia de la licencia y el copyright. Las modificaciones realizadas se deben hacer bajo la misma licencia y los cambios en el código tienen que ser documentados. Respecto a las limitaciones son las mismas que para la licencia MIT: limitación de responsabilidad y no asunción de garantías. Finalmente, la licencia GPLv2 (GNU General Public License v2.0) es similar a la que acabamos de describir, con la única diferencia de que no incluye el permiso de los derechos de la patente.

7.3 Elección de licencia para la toolbox

La licencia que se va a elegir para esta toolbox es la licencia MIT. Por todas las características descritas anteriormente ya que lo que más nos interesa es que el código sirva para el mayor número de desarrolladores y aporten contribuciones con la condición de que se incluyan los derechos de autor [23].

Capítulo 8. Conclusiones

Las grandes cantidades de datos generadas al día por las personas, dispositivos móviles etc. pueden contener información relevante o no y en muchas ocasiones ser difícil de manejar. Por ello podemos ver cómo Big Data y en concreto las técnicas de Aprendizaje Máquina son un medio idóneo para ese manejo y procesado de los datos, a fin de extraer la información de interés.

Dentro del Aprendizaje Máquina el presente trabajo se ha centrado en el Análisis Multivariable, logrando implementar una toolbox sobre Spark para la obtención de la solución sugerida en [9] paralelizando buena parte de los cálculos necesarios. El trabajo experimental llevado a cabo ha dejado fuera de su alcance el análisis de las prestaciones de los métodos. Sin embargo, mediante una serie de experimentos, sí se ha corroborado que la implementación verifica las propiedades teóricas de los métodos, validando además su convergencia a la solución centralizada en ausencia de regularización. Como se ha podido comprobar la elección de las herramientas ha sido la más adecuada ya que hoy en día Spark es el framework más potente por sus librerías y escalabilidad para el desarrollo de soluciones Big Data.

Por otro lado el impacto de este proyecto se enmarca dentro del impacto a nivel social y económico que tiene el Big Data, con el objetivo de formar parte de soluciones más complejas siendo así un proyecto Open Source.

Este proyecto no termina aquí, ya que las librerías empleadas están en constante modificación y desarrollo. Por ello como trabajos futuros, una

modificación inmediata es la migración a DataFrames debido a la importancia que ha cobrado esta abstracción frente a los RDDs y a sus ventajas.

No sólo se contempla un cambio de abstracción si no también la revisión continúa de los algoritmos empleados en el método ya que muchos de ellos en versiones futuras de Spark podrían eliminarse. También la creación de nuevas funcionalidades y la contemplación de más formatos de entrada podría tenerse en cuenta con el fin de proporcionar una toolbox lo más completa posible.

Bibliografía

[1] D. Mysore, S. Khupat y S. Jain, «IBM,» 17 9 2013. [En línea]. Available: https://www.ibm.com/developerworks/ssa/library/bd-archpatterns1/. [Último acceso: 2 2017].

[2] J. L. Aguilar y E. Leiss, Introducción a la Computación Paralela, Mérida (Venezuela): Universidad de los Andes, 2004.

[3] S. Wadkar, M. Siddalingaiah y J. Venner, Pro Apache Hadoop, Apress, 2014.

[4] Tutorials point, «Tutorials point,» 2013. [En línea]. Available: https://www.tutorialspoint.com/map_reduce/map_reduce_introduction.htm. [Último acceso: Marzo 2017].

[5] C. Bishop, Pattern Recognition and Machine Learning, Cambridge: Springer, 2010.

[6] J. Shlens, «A Tutorial on Principal Component Analysis,» p. 12, 2014. Available: https://arxiv.org/pdf/1404.1100.pdf

[7] M. Borga, «Canonical Correlation, a Tutorial,» p. 12, 2001. Avilable: http://www.imt.liu.se/~magnus/cca/tutorial/tutorial.pdf

[8] K. Siong, «A Simple Explanation of Partial Least Squares,» p. 10, 2013. Available: http://users.cecs.anu.edu.au/~kee/pls.pdf

[9] S. Muñoz-Romero, V. Gómez-Verdejo y J. Arenas-García, «Why (and How) Avoid Orthogonal Procustes in Regularized Multivariate Analysis,» p. 9, 2016. Available: https://arxiv.org/pdf/1605.02674.pdf

[10] H. Karau, A. Konwinski, P. Wendell y M. Zaharia, Learning Spark, Sebastopol: O'Reilly Media, 2015.

[11] Girardot, «Wordpress,» Mayo 2015. [En línea]. Available: https://ogirardot.wordpress.com/2015/05/29/rdds-are-the-new-bytecode-of-apache-spark/. [Último acceso: Mayo 2017].

[12] R. Xin, M. Armbrust y D. Liu, «Databricks,» Febrero 2015. [En línea]. Available: https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html. [Último acceso: Mayo 2017].

[13] Apache Spark, «Apache Spark,» [En línea]. Available: https://spark.apache.org/docs/latest/ml-guide.html. [Último acceso: Marzo 2017].

[14] Apache Spark, «Apache Spark,» [En línea]. Available: http://spark.apache.org/docs/2.1.0/api/python/pyspark.html. [Último acceso: Junio 2017].

[15] H. Zou y T. Hastie, «Stanford University,» [En línea]. Available: http://web.stanford.edu/~hastie/TALKS/enet_talk.pdf. [Último acceso: Mayo 2017].

[16] Apache Spark, «Apache Spark,» [En línea]. Available: https://spark.apache.org/docs/2.1.0/mllib-data-types.html#labeled-point. [Último acceso: Mayo 2017].

[17] Apache Spark, «Apache Spark,» [En línea]. Available: https://spark.apache.org/docs/2.1.0/api/python/pyspark.mllib.html#pyspark.mllib.linalg.Vectors. [Último acceso: Mayo 2017].

[18] M. Velasco, «Las API como multipliacodoras del software cívico: nuevos impactos para el crowfunding,» *El diario,* Mayo 2015. Avilable: http://www.eldiario.es/colaboratorio/API-multiplicadoras-software-impactos-crowdfunding_6_392020826.html

[19] E. Merino, «La explotación de Big Data en Europa supondrá la creación de 3.7 millones de empleos,» *Economía 3,* Abril 2015. Available: http://www.economia3.com/2015/04/21/47411-la-explotacion-de-big-data-en-europa-supondra-la-creacion-de-37-millones-de-empleos/

[20] F. Deed y J. McHugh, Open Source Technology and Policy, Cambridge University Press, 20008.

[21] R. Stallman, Software libre para una sociedad libre, Traficantes de sueños, 2007.

[22] B. Balter, «Open source license usage on GitHub.com,» Marzo 2015. [En línea]. Available: https://github.com/blog/1964-open-source-license-usage-on-github-com. [Último acceso: Junio 2017].

[23] GitHub, «GitHub,» [En línea]. Available: https://choosealicense.com/licenses/mit/. [Último acceso: Junio 2017].

[24] Instituto de Marketing Online, «Educación online,» Abril 2016. [En línea]. [Último acceso: Mayo 2017].Avilable: http://www.educacionline.com/instituto-de-marketing-online/que-pasa-en-un-solo-minuto-de-2016-en-internet/

Anexo I. Resumen en inglés

1. INTRODUCTION

The main goal of this project is to focus on the Big Data technologies in particular, Machine Learning and Multivariate Analysis (MVA). Many of the datasheets have a great number of variables in order to measure all the things that can influence the data. A lot of times this variables do not contribute with information to the set and are correlated, for that reason it is interesting to eliminate it and create a new datasheet. The dimension is reduced, so we achieve quick processed and clearer visualizations. The objective is to create a toolbox which gives solution to that problem with the methods already described in the literature.

2. DESCRIPTION OF THE PROBLEM

Nowadays Big Data is a wide term which refers to the type of data to process, the storage systems or the definition of the area in charge to the analysis of massive data. Big Data is related to programming models and distributed systems due to the search of speed in the processing and results. In order to give a complete definition of Big Data, in the literature it is described with the five V: Volume, Velocity, Variety, Variability and Value. We can approximate the architecture of a data analysis system for a Big Data architecture where we can differentiate five principal components: collection of data, storage, processing,

analysis and visualization. With the execution of these five elements we accomplish the management of a full problem of Big Data.

The way in which we can achieve the required speed processing large datasheets of data is by means of the parallel computation. Parallel computation is defined by the division of one task in more little tasks which are executed at the same time. This computation can be done with the help of some paradigms, the main of them, the MapReduce paradigm. It consists of two parts a Map and a Reduce function which are executed in the distributed nodes.

Machine Learning is a branch of knowledge of Artificial Intelligence. It tries to provide to the computers the abiUlity of learning without have been programmated and act according to that learnt information. Some of the applications of Machine Learning are: prediction, data mining, image recognition, behavior analysis and much more. This learning can be done with supervised or unsupervised algorithms, with the first ones we have fitted previously a training datasheet and with the second ones we do not have previous data so we can only find patterns. The principal problems that we can solve are Classification and Regression.

The multivariate analysis holds the analysis of datasheets with multiple variables which belongs to the same object in study. It has methods in order to extract relevant features for the datasheet, the main ones we are going to study are: PCA, CCA and OPLS.

The problem which is going to be developed is based on the paper "*Why (and How) Avoid Orthogonal Procustes in Regularized Mutivariate Analysis"* [9].

This theoretical solution appears due to the last contributions which ignore the basic property of the multivariate methods: the uncorrelation of the extracted features. In summary this method consists on two steps: step U (a basic problem of least squares) and step W (eigenvalue problem). With these methods we achieve that the learning tasks are easen because the least square problem can work independently over each dimension and the effects of variation are isolated in each dimension. The selection of the optimal subsequent is reduced to select the features with highest associated eigenvalue.

3. DESIGN OF THE TECHNICAL SOLUTION

For the implementation of the technical solution I have used the website Databricks due to the fact that it provides a scalable cluster for Apache Spark and notebooks for Python, the programming language chosen for the code. Apache Spark is an Open Source framework of parallel processing in clusters. The speed is achieved executing in distributed memory, Spark also extends the MapReduce model and has APIs for Java, Scala and Python. Its principal units are Spark SQL, GraphX and MLlib (the library of Machine Learning algorithms). The basic abstraction for Spark is called RDD (Resilient Distributed Dataset), it represents a collection of elements in distributed memory and partitions. These collections are immutable, when transformations are done with them, new RDDs are being created. Also is important to remark a new abstraction in the MLlib API, DataFrames. A DataFrame is a collection of distributed data organized by columns. Now the API of DataFrames is the principal API of Machine Learning because it provides a great number of advantages.

The libraries which have been used in the toolbox design are MLlib with the abstraction of RDDs which provides algorithms for Classification, Regression, Decission Trees, Clustering, Topic Modeling etc. The Python library Numpy which operates internally with Python in Spark and provides the fundamental package of multidimensional objects, linear algebra functions and so on. Scikit-Learn also an Open Source library of Machine Learning for Python it offers tools for data mining and data analysis as MLlib.

Some statistical properties for the code that can be highlighted are the regularization and the normalization. The regularization is important because it introduces additional information in order to avoid the overffiting of the model which is not desire since we can lose the predictive capacity because the model is not going to react in presence of variations in the data. The regularizations that support the toolbox are the same of the regularizations supported by MLlib this are L1, L2 norms and the elastic-net. The choice of the regularization parameter is also important since it trades the importance of the regularization. In refence to the normalization, it is used the one which is provided by MLlib, the Gaussian with mean zero and unit variance.

The toolbox is mainly designed for problems of classification and regression so that it is important to take into account the structure of the data. The type of data that the toolbox support are a RDD of Labeled point, RDD of tuples of NumPy arrays where the first element is the input data and the second one the output data (a label for classification or a NumPy array for regression) and in case that the type of MVA chosen is PCA a RDD of NumPy arrays since X=Y.

The class of the toolbox has the following functions:

- *init()* which is used to instantiate the object and the arguments it needs are: the type of MVA (typeMVA), the type of regularization (typeReg), if the user wants to normalize or not (typeNorm), a value for the stop condition for the algorithm (tol), the number of features to extract (numFeatures), the regularization parameter (regParam) by default 0.01, the step for the SGD algorithm (step) by default 1e-3, the number of iterations for SGD (iterations) by default 100 and the maximum number of U steps (max_Ustep) by default 10.
- *prepareData()* which make some computations with the input data for the correct manipulation of the collection.
- *calcCov()* computes the covariance matrix needed for others functions.
- *createOmega()* creates a matrix specific for each MVA type.
- *normalizer()* normalizes the data if it is needed.
- *stepU()* makes the step U previous defined in order to create a U matrix. It is executed iteratively with the stepW().
- *stepW()* makes the step W previous defined and create de matrix W needed for stepU().
- *computeMSE()* which computes the minimum square error
- *fit()* which fits the model doing the iteratively execution between *stepU()* and *stepW()* until the stop condition is reached
- *predict()* returns the new RDD with the selected extracted features

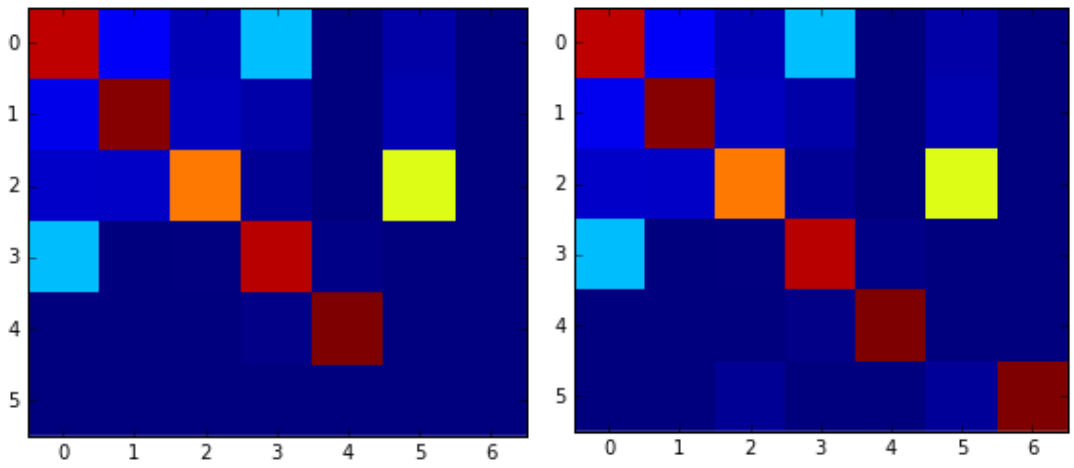
The functional description basically consists on creating the object of the MVA class with the values of the parameters we want. We just have to fit the model calling the function *fit()* and the next step is to call the function *predict()* in order to transform the input data.

4. EVALUATION

In order to validate the implemented software it will be used two types of data: synthetic data and a real datasheet.

The synthetic data has been designed on purpose of giving more relevance to the four first variables of the input data. For the output data, the regression coefficients have been designed in order to provide more relevance to the fifth variable.

The first proof which is going to do is that the PCA solution of the toolbox's method is equivalent to the solution obtained with Scikit Learn. In Figure 17 we can verify that the solution is the same in (a) for the PCA of the toolbox and in (b) PCA of Scikit-Learn.



(a) (b)

Figura 17. PCA results

Now we proof the capacity of OPLS to indentify the correlated variable with the output data. In Figure 18 we can see what we expected that the greater eigenvalue is associated to the projection of the fifth variable.

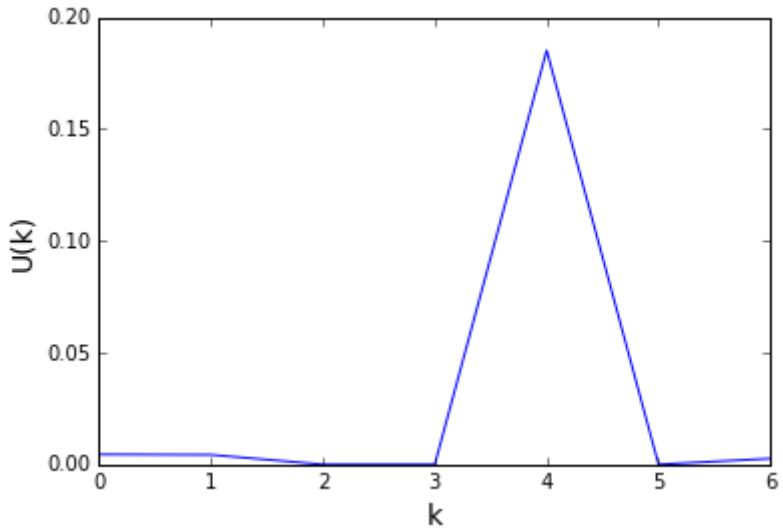


Figura 18. OPLS projection

The next objective is to visualise how the coefficients of the U matrix starts to decrease while the regularization term is increasing. In Figure 19 we can see this effect for regularization terms 0.1, 0.01 and 0.0001 respectively.

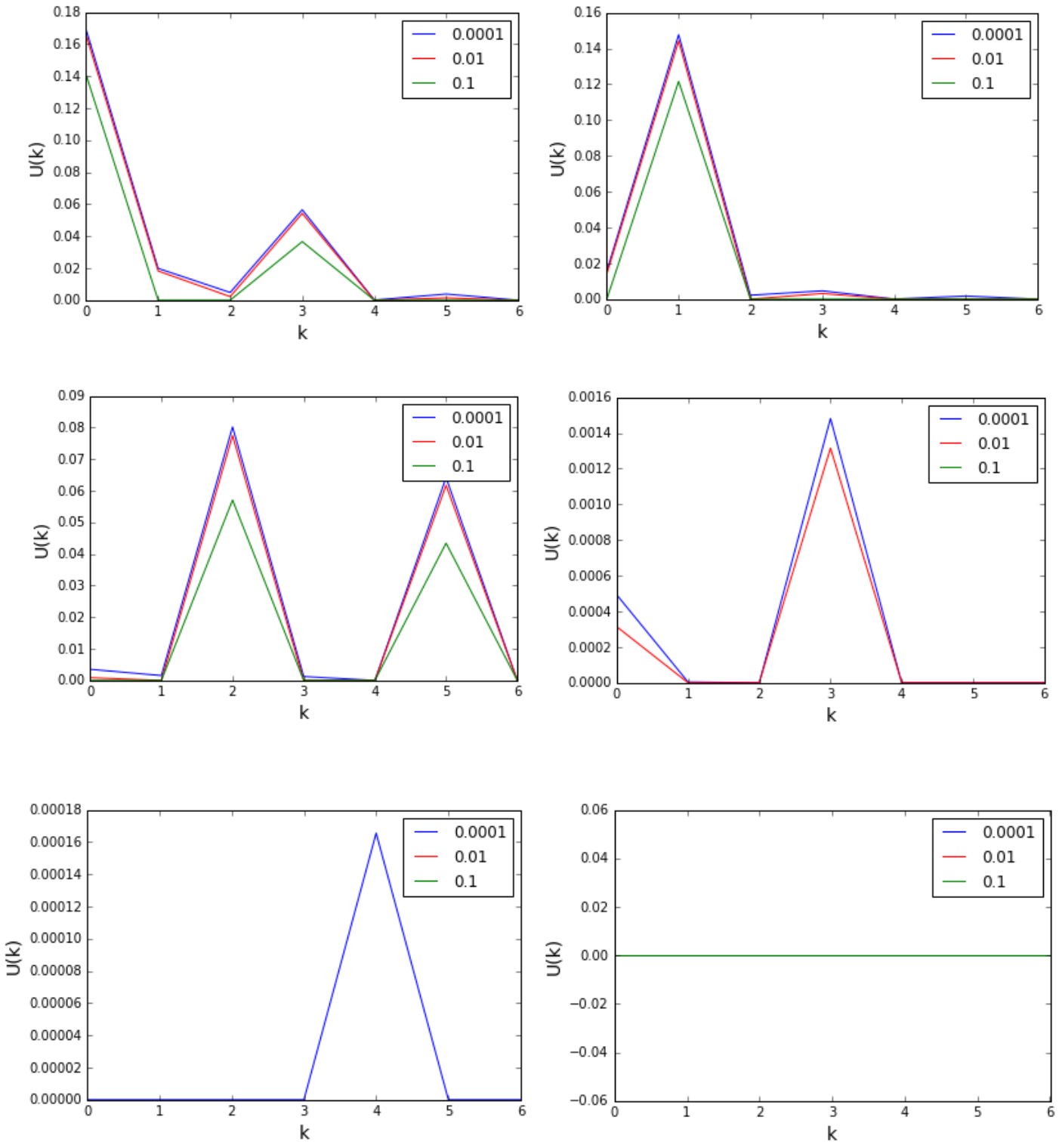


Figura 19. Projection vectors

In order to study the whitened of the projected data while the regularization term is increasing, we are going to sum all the elements in absolute value without the elements which belong to the diagonal. The chosen regularization terms were 0.00001, 0.001, 0.1 and 0.5. For these values we the sums represented in the Figure 20.

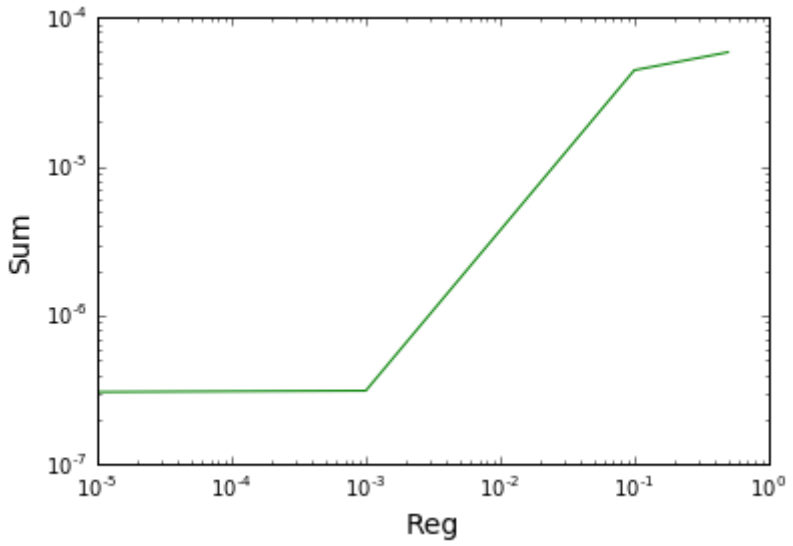


Figura 20. Sum of the matrix's elements according to the regularization term

The real datasheet used consist of 4435 samples, each of them corresponding to a pixel values in a satellite image. The purpose of this datasheet is for a classification problem so it has 36 attributes and a label.

With this datasheet we are going to evaluate the scalability of the method. We are going to measure the time which is taken in order to fit the model with 3 partitions of the input data: 1%, 10% and 100%. In the Figure 21 we can see the linear growth of the time.

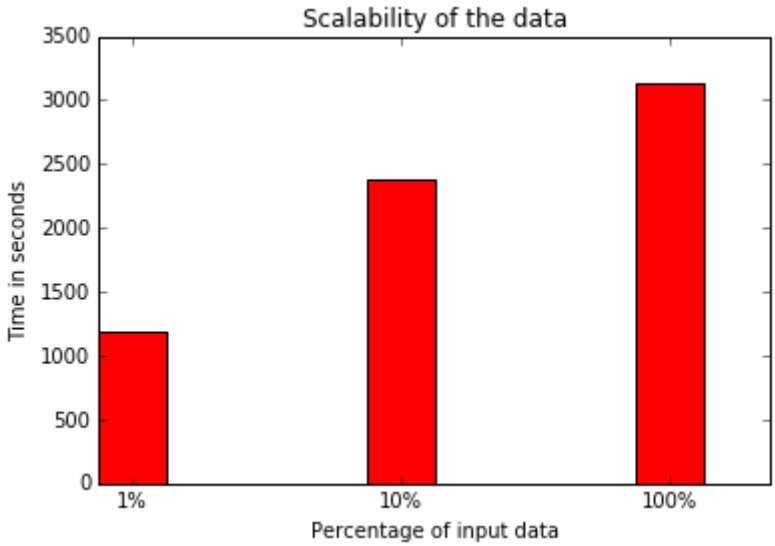


Figura 21. Scalability of the input data

The last proof is to obtain the objective function depending on the number of chosen projections. It means, to calculate the minimum squared error (MSE) depending on the number of projections taken. In Figure 22 we can see how it decreases when the projections increases.

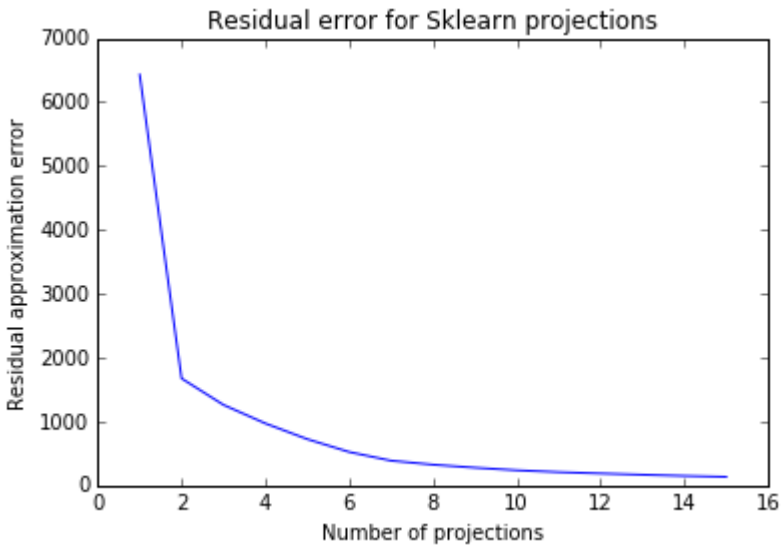


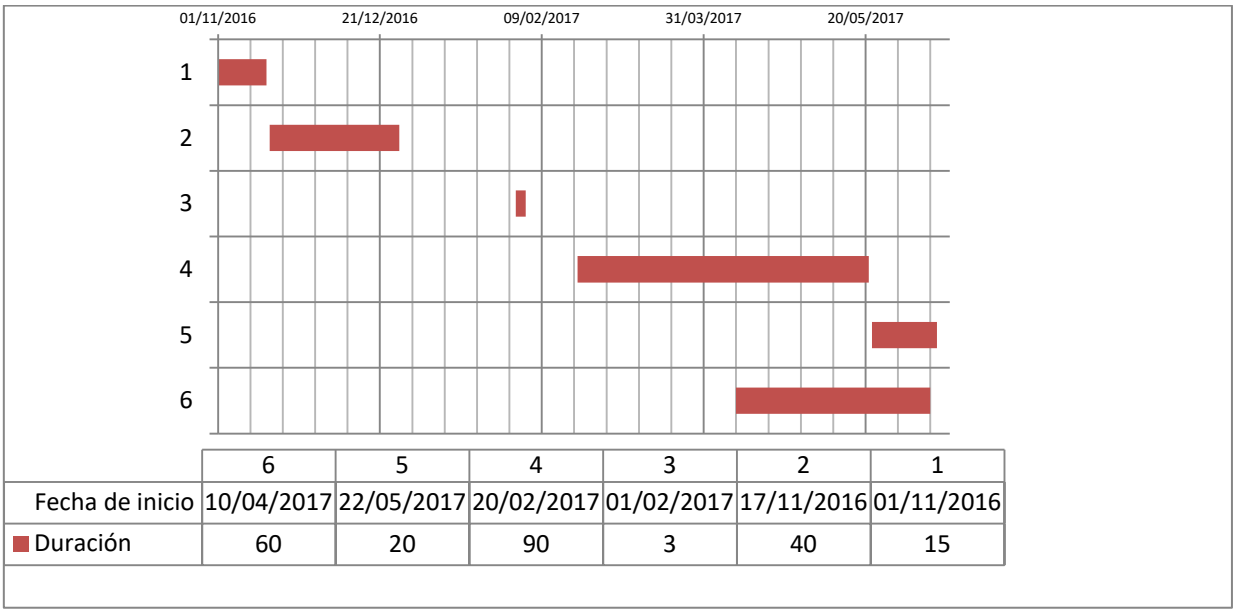
Figura 22. MSE according to the number of projections

5. PROYECT PLANING AND BUDGET

The development of the project can be summarized in six activities which are:

1. Learning Python
2. Realization of MOOC about Big Data with Apache Spark
3. Understanding the code provided by the department
4. Implementation of the toolbox code
5. Realization of the pertinent proofs in order to test the functionalities
6. Writing the memory of the project

In the Figure 23 we can see the Gantt diagram associated to the project's activities.



	6	5	4	3	2	1
Fecha de inicio	10/04/2017	22/05/2017	20/02/2017	01/02/2017	17/11/2016	01/11/2016
Duración	60	20	90	3	40	15

Figura 23. Gantt Diagram

For the estimation of the budget it has been taking into account the costs of the software and hardware used and the cost of the staff involve. For the cost of the staff it was considered an average of 2 hours of work per day of dedication with a cost of 10 euro per hour. For the final cost of software and hardware

resources, it was taken into account the amortization and the duration of the resources' usage. The considered resources are a computer, Databricks and a computational node. With this, in the Table 7 it is represented the total cost of the project.

Description	Cost
Staff cost	4560€
Resources cost	191.1€
Total	4751.1€

Tabla 7. Total cost of the project

6. SOCIO-ECONOMIC IMPACT

The social impact is based on the impact that supposes for the current digital society the Free Software. The technology generated with open code is useful for other users due to the opportunity to generate knowledge doing modifications, sharing the new results obtained and so on. Nowadays this vision has allowed the existence of an amount of free software in the context of Big Data. Currently, big companies are benefiting the potential of the volumes of data and the libraries in open code which manipulate this data for example Scikit Learn. These projects create value and settle the basis of a new way to generate knowledge, expanded it. In particular with the MVA toolbox it is pretended to supply to the users an effective tool of extracted features to reduce the dimension of the data. It is easy to see the possibilities of integration in other applications which demand the filtration of a subset. This toolbox has been created for the accessibility and for the desire of belonging to a common project.

In relation to the economic impact it is difficult to estimate it by its own. However, as long as this project is focus on bigger projects related Big Data and Machine Learning it can be measure the economic impact that this two terms have. In the next years one of the jobs in demand will be data analyst. The companies have changed its mind noticing about the relevance of a good interpretation of the data. A great number of public sectors have been benefited of the technologies of Big Data for example public health or tourism.

7. REGULATORY FRAMEWORK

The code for the Toolbox will be available en GitHub, this platform works with the version control system Git. The use of these platforms has been increasing inside the Open Source community. The term Open Source makes reference to the distributed and free development software where the code can be shared and modified in order to improve it. The main difference between the Open Source and Free Software is that Free Software emphasize ethical questions of freedom while for Open Source worries about the benefit of sharing the code. The way in which is regulated is by means of licenses. The chosen license for the Toolbox is the MIT license. Among the permissions this license has we can find the permission of commercial use, distribution, modification and private use. It has the conditions of preservation of the copyright and license notices. Also it has the limitations of liability and warranty.

8. CONCLUSION

The amount of data generated per day sometimes can contain relevant information or not and difficult to manage. Therefore we can see how the Big

Data and the technique of Machine Learning are suitable for the manage and processed of data.

This project has achieved the implementation of a solution for the techniques of feature extraction in Multivariate Analysis, focusing on corroborating the functionality of the toolbox not analysing the benefits of the proposed method.

The future works for this project are immediate due to the constant modification and development of the libraries used. The first modification will be the migration to DataFrames and afterwards the introduction of new functionalities in order to provide a complete toolbox.